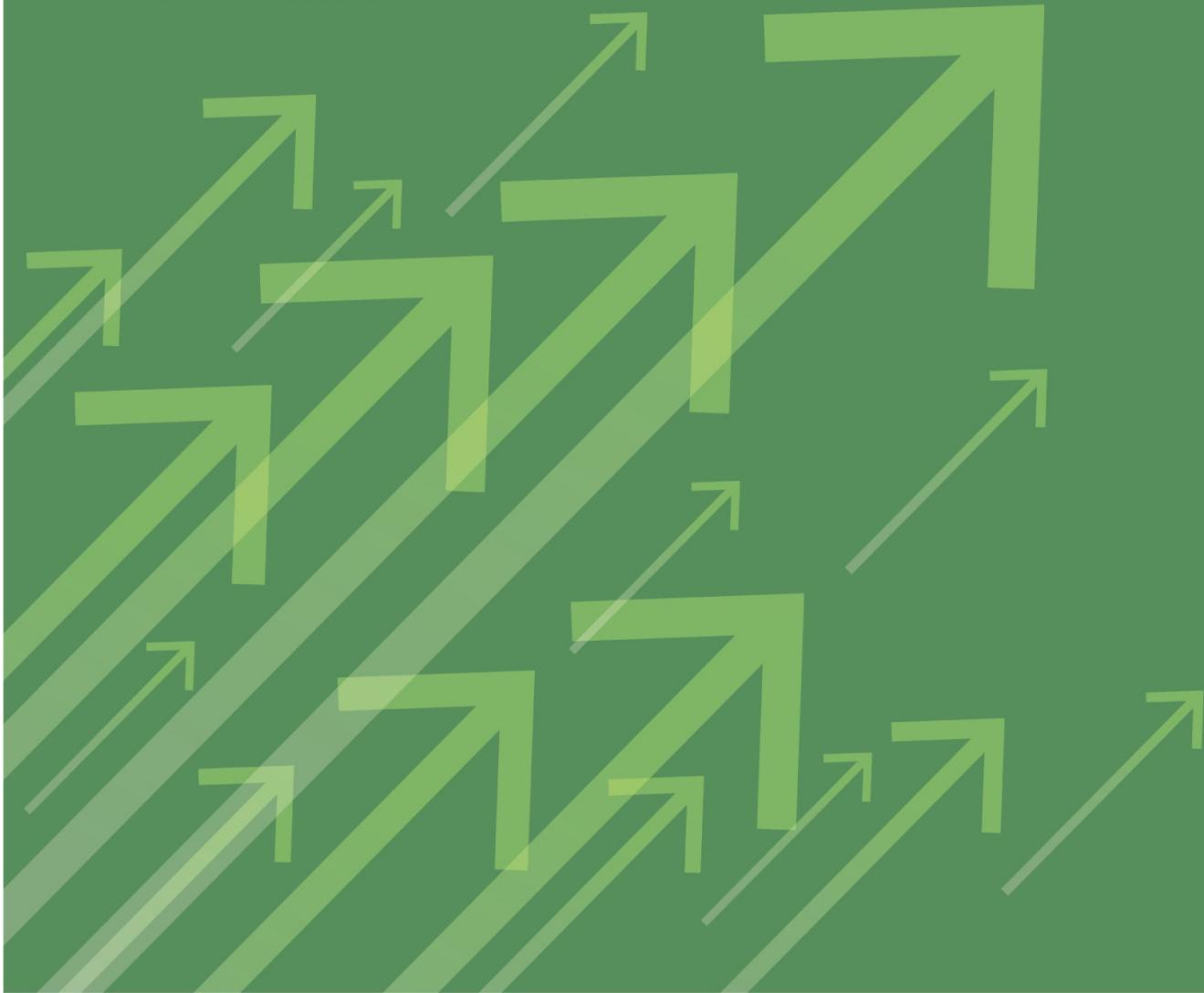


Microeconometrics Using Stata

Volume II: Nonlinear Models and
Causal Inference Methods

Second Edition



A. COLIN CAMERON
PRAVIN K. TRIVEDI

STATA
Press

Microeometrics Using Stata

Volume II: Nonlinear Models and
Causal Inference Methods

Second Edition



A. COLIN CAMERON
PRAVIN K. TRIVEDI

STATA
Press

Microeometrics Using Stata

Volume II: Nonlinear Models and Causal Inference 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

16 [Nonlinear optimization methods](#)

- 16.1 [Introduction](#)
- 16.2 [Newton–Raphson method](#)
- 16.3 [Gradient methods](#)
- 16.4 [Overview of ml, moptimize\(\), and optimize\(\)](#)
- 16.5 [The ml command: lf method](#)
- 16.6 [Checking the program](#)
- 16.7 [The ml command: lf0–lf2, d0–d2, and gf0 methods](#)
- 16.8 [Nonlinear instrumental-variables \(GMM\) example](#)
- 16.9 [Additional resources](#)
- 16.10 [Exercises](#)

17 [Binary outcome models](#)

- 17.1 [Introduction](#)
- 17.2 [Some parametric models](#)
- 17.3 [Estimation](#)
- 17.4 [Example](#)
- 17.5 [Goodness of fit and prediction](#)
- 17.6 [Marginal effects](#)
- 17.7 [Clustered data](#)
- 17.8 [Additional models](#)
- 17.9 [Endogenous regressors](#)
- 17.10 [Grouped and fractional data](#)
- 17.11 [Additional resources](#)
- 17.12 [Exercises](#)

18 [Multinomial models](#)

- 18.1 [Introduction](#)
- 18.2 [Multinomial models overview](#)
- 18.3 [Multinomial example: Choice of fishing mode](#)
- 18.4 [Multinomial logit model](#)
- 18.5 [Alternative-specific conditional logit model](#)
- 18.6 [Nested logit model](#)

- 18.7 [Multinomial probit model](#)
- 18.8 [Alternative-specific random-parameters logit](#)
- 18.9 [Ordered outcome models](#)
- 18.10 [Clustered data](#)
- 18.11 [Multivariate outcomes](#)
- 18.12 [Additional resources](#)
- 18.13 [Exercises](#)

19 [Tobit and selection models](#)

- 19.1 [Introduction](#)
- 19.2 [Tobit model](#)
- 19.3 [Tobit model example](#)
- 19.4 [Tobit for lognormal data](#)
- 19.5 [Two-part model in logs](#)
- 19.6 [Selection models](#)
- 19.7 [Nonnormal models of selection](#)
- 19.8 [Prediction from models with outcome in logs](#)
- 19.9 [Endogenous regressors](#)
- 19.10 [Missing data](#)
- 19.11 [Panel attrition](#)
- 19.12 [Additional resources](#)
- 19.13 [Exercises](#)

20 [Count-data models](#)

- 20.1 [Introduction](#)
- 20.2 [Modeling strategies for count data](#)
- 20.3 [Poisson and negative binomial models](#)
- 20.4 [Hurdle model](#)
- 20.5 [Finite-mixture models](#)
- 20.6 [Zero-inflated models](#)
- 20.7 [Endogenous regressors](#)
- 20.8 [Clustered data](#)
- 20.9 [Quantile regression for count data](#)
- 20.10 [Additional resources](#)
- 20.11 [Exercises](#)

21 [Survival analysis for duration data](#)

- 21.1 [Introduction](#)

- 21.2 [Data and data summary](#)
- 21.3 [Survivor and hazard functions](#)
- 21.4 [Semiparametric regression model](#)
- 21.5 [Fully parametric regression models](#)
- 21.6 [Multiple-records data](#)
- 21.7 [Discrete-time hazards logit model](#)
- 21.8 [Time-varying regressors](#)
- 21.9 [Clustered data](#)
- 21.10 [Additional resources](#)
- 21.11 [Exercises](#)

22 Nonlinear panel models

- 22.1 [Introduction](#)
- 22.2 [Nonlinear panel-data overview](#)
- 22.3 [Nonlinear panel-data example](#)
- 22.4 [Binary outcome and ordered outcome models](#)
- 22.5 [Tobit and interval-data models](#)
- 22.6 [Count-data models](#)
- 22.7 [Panel quantile regression](#)
- 22.8 [Endogenous regressors in nonlinear panel models](#)
- 22.9 [Additional resources](#)
- 22.10 [Exercises](#)

23 Parametric models for heterogeneity and endogeneity

- 23.1 [Introduction](#)
- 23.2 [Finite mixtures and unobserved heterogeneity](#)
- 23.3 [Empirical examples of FMMs](#)
- 23.4 [Nonlinear mixed-effects models](#)
- 23.5 [Linear structural equation models](#)
- 23.6 [Generalized structural equation models](#)
- 23.7 [ERM commands for endogeneity and selection](#)
- 23.8 [Additional resources](#)
- 23.9 [Exercises](#)

24 Randomized control trials and exogenous treatment effects

- 24.1 [Introduction](#)
- 24.2 [Potential outcomes](#)
- 24.3 [Randomized control trials](#)

- 24.4 [Regression in an RCT](#)
- 24.5 [Treatment evaluation with exogenous treatment](#)
- 24.6 [Treatment evaluation methods and estimators](#)
- 24.7 [Stata commands for treatment evaluation](#)
- 24.8 [Oregon Health Insurance Experiment example](#)
- 24.9 [Treatment-effect estimates using the OHIE data](#)
- 24.10 [Multilevel treatment effects](#)
- 24.11 [Conditional quantile TEs](#)
- 24.12 [Additional resources](#)
- 24.13 [Exercises](#)

[25 Endogenous treatment effects](#)

- 25.1 [Introduction](#)
- 25.2 [Parametric methods for endogenous treatment](#)
- 25.3 [ERM commands for endogenous treatment](#)
- 25.4 [ET commands for binary endogenous treatment](#)
- 25.5 [The LATE estimator for heterogeneous effects](#)
- 25.6 [Difference-in-differences and synthetic control](#)
- 25.7 [Regression discontinuity design](#)
- 25.8 [Conditional quantile regression with endogenous regressors](#)
- 25.9 [Unconditional quantiles](#)
- 25.10 [Additional resources](#)
- 25.11 [Exercises](#)

[26 Spatial regression](#)

- 26.1 [Introduction](#)
- 26.2 [Overview of spatial regression models](#)
- 26.3 [Geospatial data](#)
- 26.4 [The spatial weighting matrix](#)
- 26.5 [OLS regression and test for spatial correlation](#)
- 26.6 [Spatial dependence in the error](#)
- 26.7 [Spatial autocorrelation regression models](#)
- 26.8 [Spatial instrumental variables](#)
- 26.9 [Spatial panel-data models](#)
- 26.10 [Additional resources](#)
- 26.11 [Exercises](#)

[27 Semiparametric regression](#)

- 27.1 [Introduction](#)
- 27.2 [Kernel regression](#)
- 27.3 [Series regression](#)
- 27.4 [Nonparametric single regressor example](#)
- 27.5 [Nonparametric multiple regressor example](#)
- 27.6 [Partial linear model](#)
- 27.7 [Single-index model](#)
- 27.8 [Generalized additive models](#)
- 27.9 [Additional resources](#)
- 27.10 [Exercises](#)

[28 Machine learning for prediction and inference](#)

- 28.1 [Introduction](#)
- 28.2 [Measuring the predictive ability of a model](#)
- 28.3 [Shrinkage estimators](#)
- 28.4 [Prediction using lasso, ridge, and elasticnet](#)
- 28.5 [Dimension reduction](#)
- 28.6 [Machine learning methods for prediction](#)
- 28.7 [Prediction application](#)
- 28.8 [Machine learning for inference in partial linear model](#)
- 28.9 [Machine learning for inference in other models](#)
- 28.10 [Additional resources](#)
- 28.11 [Exercises](#)

[29 Bayesian methods: Basics](#)

- 29.1 [Introduction](#)
- 29.2 [Bayesian introductory example](#)
- 29.3 [Bayesian methods overview](#)
- 29.4 [An i.i.d. example](#)
- 29.5 [Linear regression](#)
- 29.6 [A linear regression example](#)
- 29.7 [Modifying the MH algorithm](#)
- 29.8 [RE model](#)
- 29.9 [Bayesian model selection](#)
- 29.10 [Bayesian prediction](#)
- 29.11 [Probit example](#)
- 29.12 [Additional resources](#)
- 29.13 [Exercises](#)

30 Bayesian methods: Markov chain Monte Carlo algorithms

- 30.1 [Introduction](#)
- 30.2 [User-provided log likelihood](#)
- 30.3 [MH algorithm in Mata](#)
- 30.4 [Data augmentation and the Gibbs sampler in Mata](#)
- 30.5 [Multiple imputation](#)
- 30.6 [Multiple-imputation example](#)
- 30.7 [Additional resources](#)
- 30.8 [Exercises](#)

[**Glossary of abbreviations**](#)

[**References**](#)

[**Author index**](#)

[**Subject index**](#)

Tables

[16.1 Evaluator types for `ml`, `moptimize\(\)`, and `optimize\(\)`](#)

[16.2 Optimization techniques for evaluator types for `ml`, `moptimize\(\)`, and `optimize\(\)`](#)

[17.1 Four commonly used binary outcome models](#)

[18.1 Commands for multinomial models](#)

[19.1 Moments of normal errors truncated from above](#)

[19.2 Some standard copula functions](#)

[19.3 Conditional and unconditional means for models in logs](#)

[20.1 Goodness-of-fit criteria for six models](#)

[22.1 Commands for nonlinear panel models](#)

[23.1 `sem` commands and equivalent standard Stata commands](#)

[23.2 `gsem` commands and equivalent standard Stata commands](#)

[23.3 Summary of ERM commands and command options](#)

[23.4 Selected ERM commands for handling endogeneity and selection](#)

[24.1 Size and power analysis terminology and notation](#)

[24.2 Key basic TES methods and commands](#)

[24.3 `teffectsra` and ERM commands when treatment is exogenous](#)

[25.1 Selected ERM commands when treatment is endogenous](#)

[25.2 LATE: The four types of individuals](#)

[25.3 Selected options of `rdrobust` command](#)

Figures

- [16.1 Objective function with multiple optimums](#)
- [17.1 Predicted probabilities versus `hhincome` and receiver operator characteristics curve following `logit`](#)
- [17.2 Predicted probabilities from local `logit` compared with `logit ML`](#)
- [19.1 Log earnings by attrition status for wave 1 and for wave 3](#)
- [20.1 Four count distributions with the same mean](#)
- [20.2 Fitted values distribution, `FMM2-P`](#)
- [20.3 Densities of posterior probabilities for `FMM Poisson` and `NB2`](#)
- [20.4 Quantile plots of count `docvis` \(left\) and its jittered transform \(right\)](#)
- [21.1 Histograms for complete and incomplete spells](#)
- [21.2 Kaplan–Meier estimate of survivor function](#)
- [21.3 Kaplan–Meier estimate of survivor function](#)
- [21.4 Survivor and related curves following `Cox PH regression`](#)
- [21.5 Check for parallel log–log survivor curves](#)
- [21.6 Compare `Cox PH` to Kaplan–Meier survivor curve](#)
- [21.7 Graph of Schoenfeld residuals against spell length](#)
- [21.8 Hazard rate from various survival models](#)
- [21.9 Weibull hazard controlling for frailty](#)
- [23.1 Fitted means from `FMM2 gamma regression`](#)
- [23.2 Posterior probabilities from `FMM2 gamma regression`](#)
- [23.3 Fitted probabilities from `FMM2 logit regression`](#)
- [23.4 Path diagram for measurement-error model](#)
- [23.5 Path diagram for regression with measurement error](#)
- [24.1 Quantile plots of `TE`](#)
- [24.2 Propensity-score overlap](#)
- [25.1 Synthetic control: Outcomes and difference in outcomes](#)
- [25.2 Synthetic control: Treated and donor outcomes](#)
- [25.3 SRD: Scatterplot and global or local regression fits](#)
- [25.4 RD: Binned data scatterplot and check of discontinuity in the running variable](#)
- [25.5 SRD: Scatterplot and binned plot](#)
- [25.6 SRD: ATE estimate using local linear and local quadratic regression](#)
- [25.7 SRD: ATE estimate using local linear and local quadratic regression](#)
- [25.8 SRD: Placebo plots](#)

26.1 Heat map of homicide rate in various counties

27.1 Local linear regression of earnings on hours with bootstrap confidence intervals

27.2 Semiparametric regression: Partial linear and single-index models

27.3 GAM: Partial residuals and fitted $g(\cdot)$ against regressors x_1, x_2 , and z

28.1 Lasso versus ridge

28.2 Plots of CV values and coefficients as λ changes

29.1 Posterior densities of the three regression parameters and the error variance parameter

29.2 Diagnostics for 10,000 MH posterior draws of μ

29.3 Diagnostics for 10,000 MH posterior draws of mu

29.4 Diagnostics for posterior draws of education coefficient

29.5 Trace for all four coefficients

29.6 Density for each half of draws for all four coefficients

30.1 Diagnostics for posterior draws of education coefficient

30.2 Diagnostics for posterior draws of education coefficient

30.3 A basic scatterplot of log earnings on hours

Chapter 16

Nonlinear optimization methods

16.1 Introduction

Much of this book covers nonlinear regression models. In many cases, these can be fit using official Stata estimation commands such as `logit` and `poisson`, without knowledge of how the estimates are obtained. Then nonlinearity merely adds the complication of how to interpret the regression output.

In this chapter, we explain the iterative methods used to obtain parameter estimates. The discussion can be relevant even when an official command is used. For example, it explains why an iteration log might include output that includes messages such as `(backed up)` or `(not concave)` and whether this is reason for concern.

Additionally, we present methods to fit a model for which there is no official command. The `ml` command enables maximum likelihood (ML) estimation if, at a minimum, the log density formula is provided. This command is more generally applicable to other *m* estimators, such as the nonlinear least-squares (NLS) estimator.

Not all models can be fit using the `ml` command. A wider range of models can be fit using the Mata `moptimize()` and `optimize()` functions for optimization; these functions are part of the Mata matrix programming language and are detailed in appendix C. The optimization tools in order of complexity are `ml`, `moptimize()`, and `optimize()`. All lead ultimately to computation using `optimize()` and, for equivalently defined models, the same numerical results.

We illustrate the `ml` command methods using ML estimation of Poisson and negative binomial regression models. Application is to the same Medical Expenditure Panel Survey data example described more fully in section 10.2 and also used in chapter 13. The chapter concludes with a nonlinear generalized method of moments (GMM) example fit using the Mata `optimize()` function.

16.2 Newton–Raphson method

Estimators that maximize an objective function, such as the log likelihood, are obtained by calculating a sequence of estimates $\hat{\theta}_1, \hat{\theta}_2, \dots$ that move toward the top of the hill. Gradient methods do so by moving by an amount that is a suitable multiple of the gradient at the current estimate. A standard method is the Newton–Raphson (NR) method, which works especially well when the objective function is globally concave.

16.2.1 NR method

We consider the estimator $\hat{\theta}$ that is a local maximum to the objective function $Q(\theta)$, so $\hat{\theta}$ solves

$$\mathbf{g}(\hat{\theta}) = \mathbf{0}$$

where $\mathbf{g}(\theta) = \partial Q(\theta)/\partial\theta$ is the gradient vector. Numerical solution methods are needed when these first-order conditions do not yield an explicit solution for $\hat{\theta}$.

Let $\hat{\theta}_s$ denote the s th round estimate of θ . Then a second-order Taylor-series expansion around $\hat{\theta}_s$ approximates the objective function $Q(\theta)$ by

$$Q^*(\theta) = Q(\hat{\theta}_s) + \mathbf{g}(\hat{\theta}_s)' (\theta - \hat{\theta}_s) + \frac{1}{2} (\theta - \hat{\theta}_s)' \mathbf{H}(\hat{\theta}_s) (\theta - \hat{\theta}_s)$$

where $\mathbf{H} = \partial g(\theta)/\partial\theta' = \partial^2 Q(\theta)/\partial\theta\partial\theta'$ is the Hessian matrix. Maximizing this approximating function with respect to θ leads to

$$\partial Q^*(\theta)/\partial\theta = \mathbf{g}(\hat{\theta}_s) + \mathbf{H}(\hat{\theta}_s)(\theta - \hat{\theta}_s) = \mathbf{0}$$

Solving for θ yields the NR algorithm

$$\hat{\theta}_{s+1} = \hat{\theta}_s - \mathbf{H}(\hat{\theta}_s)^{-1} \mathbf{g}(\hat{\theta}_s) \quad (16.1)$$

The parameter estimate is changed by a matrix multiple of the gradient vector, where the multiple is minus the inverse of the Hessian matrix.

The final step in deriving (16.1) presumes that the inverse exists. If instead the Hessian is singular, then $\hat{\theta}_{s+1}$ is not uniquely defined. The Hessian may be singular for some iterations, and optimization methods have methods for still continuing the iterations. However, the Hessian must be nonsingular at the optimum. This complication does not arise if $Q(\theta)$ is globally concave because then the Hessian is negative definite at all points of evaluation. In that case, the NR method works well, with few iterations required to obtain the maximum.

16.2.2 NR method for Poisson

The Poisson model is summarized in section 13.2.2. As noted at the start of section 13.2, the fact that we are modeling a discrete random variable places no restriction on the generality of the example. Exactly the same points could be illustrated using, for example, complete spell duration data modeled using the exponential or even a nonlinear model with normally distributed errors.

For the Poisson model, the objective function, gradient, and Hessian are, respectively,

$$\begin{aligned} Q(\beta) &= \sum_{i=1}^N \{-\exp(\mathbf{x}'_i \beta) + y_i \mathbf{x}'_i \beta - \ln y_i!\} \\ g(\beta) &= \sum_{i=1}^N \{y_i - \exp(\mathbf{x}'_i \beta)\} \mathbf{x}_i \\ H(\beta) &= \sum_{i=1}^N -\exp(\mathbf{x}'_i \beta) \mathbf{x}_i \mathbf{x}'_i \end{aligned} \quad (16.2)$$

Note $H(\beta) = -\mathbf{X}'\mathbf{D}\mathbf{X}$, where \mathbf{X} is the $N \times K$ regressor matrix and $\mathbf{D} = \text{Diag}\{\exp(\mathbf{x}_i'\beta)\}$ is an $N \times N$ diagonal matrix with positive entries. It follows that if \mathbf{X} is of full rank, then $H(\beta)$ is negative definite for all β , and the objective function is globally concave. Combining (16.1) and (16.2), we see the NR iterations for the Poisson maximum-likelihood estimator (MLE) are

$$\hat{\beta}_{s+1} = \hat{\beta}_s + \left\{ \sum_{i=1}^N \exp(\mathbf{x}_i'\hat{\beta}_s) \mathbf{x}_i \mathbf{x}_i' \right\}^{-1} \times \sum_{i=1}^N \left\{ y_i - \exp(\mathbf{x}_i'\hat{\beta}_s) \right\} \mathbf{x}_i$$

16.2.3 Poisson NR example using Mata

To present an iterative method in more detail, we manually code the NR algorithm for the Poisson model, using Mata functions that are explained in appendix B. The same example is used in subsequent sections to demonstrate use of the Stata `m1` command and the Mata `optimize()` and `moptimize()` functions.

Core Mata code for Poisson NR iterations

For expositional purposes, we begin with the Mata code for the core commands to implement the NR iterative method for the Poisson.

We assume that the regressor matrix x and the dependent variable vector y have already been constructed. Iterations stop when $g(\hat{\beta}_s)'H(\hat{\beta}_s)^{-1}g(\hat{\beta}_s) < 10^{-8}$, and output provides an iteration log.

```

* Core Mata code for Poisson MLE NR iterations
mata
    p      = cols(X)           // Number of regressors
    b      = J(p, 1, .02)       // Start values are close to zero
    gHinvg = 1                 // Initialize scaled gradient
    iter   = 0                 // Initialize number of iterations
    do {
        mu    = exp(X*b)
        grad = X`*(y-mu)        // k x 1 gradient vector
        hes  = cross(X, mu, X)  // Negative of the k x k Hessian matrix
        diff = cholinv(hes)*grad // Update amount
        bold = b
        b    = bold + diff
        iter = iter + 1
        gHinvg = grad`*diff     // Equals grad`*cholinv(hes)*grad
        printf("iter = %s, gHinv = %6.5g\n", strofreal(iter), gHinvg)
    } while (gHinvg > 1e-8)    // End of iteration loops
end

```

The $N \times 1$ vector `mu` has the i th entry $\mu_i = \exp(\mathbf{x}'_i \boldsymbol{\beta})$. The $K \times 1$ vector `grad` equals $\sum_{i=1}^N (y_i - \mu_i)\mathbf{x}_i$, and `hes=cross(X, mu, X)` equals $\sum_i \mu_i \mathbf{x}_i \mathbf{x}'_i$.

The `cross()` function ensures a numerical result that is a symmetric matrix. The quickest function for taking a matrix inverse is `cholinv()` for a symmetric positive-definite matrix. Thus, we set `hes` to $-H(\hat{\boldsymbol{\beta}})$, which is positive definite, but then the NR update has a plus sign rather than the minus sign in (16.1). In some cases, though not here given use of the `cross()` function, `hes` may not be symmetric because of a rounding error. Then we would add the `hes = makesymmetric(hes)` command before calling `cholinv()`.

Complete Stata and Mata code for Poisson NR iterations

The complete program has the following sequence: 1) in Stata, obtain the data, and define any macros used subsequently; 2) in Mata, calculate the parameter estimates and the estimated variance–covariance matrix of the estimator (VCE), and pass these back to Stata; and 3) in Stata, output nicely formatted results.

We begin by reading in the data and defining the global macro `y` for the dependent variable and the global macro `xlist` for the regressors.

```
. * Set up data and local macros for dependent variable and regressors
. qui use mus210mepsdocvisyoung
. keep if year02 == 1
(25,712 observations deleted)
. generate cons = 1
. global y docvis
. global xlist private chronic female income cons
```

The dependent variable is the number of office-based physician visits (`docvis`) by persons in the United States aged 25–64 years. The regressors used are health insurance status (`private`), health status (`chronic`), and socioeconomic characteristics (`female` and `income`).

The subsequent Mata program reads in the relevant data and obtains the parameter estimates and the estimate of the VCE. The program first associates vector `y` and matrix `x` with the relevant Stata variables by using the `st_view()` function. The `tokens("")` function is added to convert `$xlist` to a comma-separated list with each entry in double quotes, the necessary format for `st_view()`. The starting values are simply zero for slope parameters and one for the intercept. The robust estimate of the VCE is obtained, and this and the parameter estimates are passed back to Stata by using the `st_matrix()` function. We have

```

. * Complete Mata code for Poisson MLE NR iterations
. mata:
----- mata (type end to exit) -----
: st_view(y=., ., "$y")           // Read in stata data to y and X
: st_view(X=., ., tokens("$xlist"))
: p      = cols(X)               // Number of regressors
: n      = rows(X)
: b      = J(p, 1, .02)          // Starting values are close to zero
: gHinvg = 1                     // Initialize scaled gradient
: iter   = 0                     // Initialize number of iterations
: do {
>     mu    = exp(X*b)
>     grad = X`*(y-mu)           // k x 1 gradient vector
>     hes   = cross(X, mu, X)    // Negative of the k x k Hessian matrix
>     diff  = cholinv(hes)*grad
>     bold  = b
>     b     = bold + diff
>     iter = iter + 1
>     gHinvg = grad`*diff        // Equals grad`*cholinv(hes)*grad
>     printf("iter = %s, gHinv = %6.5g\n", strofreal(iter), gHinvg)
> } while (gHinvg > 1e-8)        // End of iteration loops
iter = 1, gHinv = 2.2e+04
iter = 2, gHinv = 1.3e+04
iter = 3, gHinv = 1648
iter = 4, gHinv = 52.48
iter = 5, gHinv = .0808
iter = 6, gHinv = 3.2e-07
iter = 7, gHinv = 8.9e-18
:     mu = exp(X*b)
:     hes = cross(X, mu, X)
:     vgrad = cross(X, (y-mu):^2, X)
:     vb = cholinv(hes)*vgrad*cholinv(hes)*n/(n-cols(X))
:     st_matrix("b",b`)           // Pass results from Mata to Stata
:     st_matrix("V",vb)           // Pass results from Mata to Stata
: end
-----
```

Once back in Stata, we use the `ereturn` command to display the results, first assigning names to the columns and rows of `b` and `V`. We have

```

. * Present results, nicely formatted using Stata command ereturn
. 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]
private	.7986654	.1090509	7.32	0.000	.5849295 1.012401
chronic	1.091865	.0560205	19.49	0.000	.9820669 1.201663
female	.4925481	.058563	8.41	0.000	.3777666 .6073295
income	.003557	.001083	3.28	0.001	.0014344 .0056796
cons	-.2297263	.1109236	-2.07	0.038	-.4471325 -.0123202

The coefficients are the same as those from the `poisson` command (see section 13.3.2), and the standard errors are the same to at least the first three significant digits. The program required seven iterations.

The preceding NR algorithm can be adapted to use Stata `matrix` commands, but it is better to use Mata functions because these can be simpler. Also, Mata functions read more like algebraic matrix expressions, and Mata does not have the restrictions on matrix size that are present in Stata.

16.3 Gradient methods

In this section, we consider various gradient methods, stopping criteria, multiple optima, and numerical derivatives. The discussion is relevant for official estimation commands, as well as for community-contributed commands.

16.3.1 Maximization options

Stata ML estimation commands, such as `poisson`, and the general-purpose `m1` command, presented in the next section, have various maximization options that are detailed in [R] **Maximize**.

The default is to provide an iteration log that gives the value of the objective function at each step plus information on the iterative method being used. This can be suppressed using the `nolog` command. Additional information at each iteration can be given by using the `trace` (current parameter values), `gradient` (current gradient vector), `hessian` (current Hessian), and `showstep` (report steps within each iteration) options.

The `technique()` option allows several maximization techniques other than NR. The `nr`, `bhhh`, `dfp`, `bfgs`, and `gn` options are briefly discussed in section [16.3.2](#).

Three stopping criteria, which are the options `tolerance(#)`, `ltolerance(#)`, and `nrtolerance(#)`, are discussed in section [16.3.4](#). The default is `nrtolerance(1e-5)`.

The `difficult` option uses an alternative method to determine steps when the estimates are in a region where the objective function is nonconcave.

The `from(init_specs)` option allows starting values to be set.

The maximum number of iterations can be set by using the `iterate(#)` option or by the separate command `set maxiter #`. The default is 300, but

this can be changed.

16.3.2 Gradient methods

Stata maximization commands use the iterative algorithm

$$\hat{\boldsymbol{\theta}}_{s+1} = \hat{\boldsymbol{\theta}}_s + a_s \mathbf{W}_s \mathbf{g}_s, \quad s = 1, \dots, S \quad (16.3)$$

where a_s is a scalar step-size adjustment and \mathbf{W}_s is a $q \times q$ weighting matrix. A special case is the NR method given in (16.1), which uses $-\mathbf{H}_s^{-1}$ in place of $a_s \mathbf{W}_s$.

If the matrix multiplier \mathbf{W}_s is too small, we will take a long time to reach the maximum, whereas if a multiple is too large, we can overshoot the maximum. The step-size adjustment a_s is used to evaluate $Q(\hat{\boldsymbol{\theta}}_{s+1})$ at $\hat{\boldsymbol{\theta}}_{s+1} = \hat{\boldsymbol{\theta}}_s + a_s \mathbf{W}_s \mathbf{g}_s$ over a range of values of a_s (such as 0.5, 1, and 2), and the value of a_s that leads to the largest value for $Q(\hat{\boldsymbol{\theta}}_{s+1})$ is chosen. This speeds up computation because calculation of $\mathbf{W}_s \mathbf{g}_s$ takes much more time than several subsequent evaluations of $Q(\hat{\boldsymbol{\theta}}_{s+1})$. It also ensures that the algorithm refuses steps that would decrease $Q(\hat{\boldsymbol{\theta}})$. Stata uses a sophisticated method to choose a_s so that convergence occurs quickly, even for difficult problems.

Different weighting matrices \mathbf{W}_s correspond to different gradient methods. Ideally, the NR method can be used, with $\mathbf{W}_s = -\mathbf{H}_s^{-1}$. If \mathbf{H}_s is nonnegative definite, noninvertible, or both, then \mathbf{H}_s is adjusted so that it is invertible. Stata also uses $\mathbf{W}_s = -\{\mathbf{H}_s + c\text{Diag}(\mathbf{H}_s)\}^{-1}$. If this fails, then Stata uses NR for the orthogonal subspace corresponding to nonproblematic eigenvalues of \mathbf{H}_s and steepest ascent ($\mathbf{W}_s = \mathbf{I}_s$) for the orthogonal subspace corresponding to problematic (negative or small positive) eigenvalues of \mathbf{H}_s .

Other optimization methods can also be used. These methods calculate alternatives to \mathbf{H}_s^{-1} that can be computationally faster and can be possible even in regions where \mathbf{H}_s is nonnegative definite, noninvertible, or both.

The alternative methods available for the `m1` command are the Berndt–Hall–Hall–Hausman, Davidon–Fletcher–Powell, and Broyden–Fletcher–Goldfarb–Shanno algorithms. These methods can be selected by specifying the `technique()` as, respectively, `bhhh`, `dfp`, and `bfgs`. Additionally, some estimators with a quadratic objective function, such as GMM, are obtained using `technique(gn)`, which uses the Gauss–Newton algorithm. The methods are explained in, for example, [Cameron and Trivedi \(2005, chap. 10\)](#), [Greene \(2018, app. E\)](#), [Wooldridge \(2010, chap. 12.7\)](#), and [Gould, Pitblado, and Poi \(2010, chap. 1\)](#).

Some of these algorithms can converge even if \mathbf{H}_s is still nonnegative definite. Then one can obtain parameter estimates but not standard errors because the latter requires inversion of the Hessian. The lack of standard errors is a clear signal of problems.

16.3.3 Messages during iterations

The iteration log can include comments on each iteration.

The message `(backed up)` is given when the original step size a_s in (16.3) resulted in a *lower* $Q(\widehat{\boldsymbol{\theta}}_{s+1})$. The message `(not concave)` means that $-\mathbf{H}_s$ was not invertible. In both cases, the ultimate results are fine, provided that these messages are not being given at the last iteration. When `(not concave)` is displayed at every iteration, however, the model is almost certainly not identified; see section [16.3.5](#).

16.3.4 Stopping criteria

The iterative process continues until it is felt that $\mathbf{g}(\widehat{\boldsymbol{\theta}}) \simeq \mathbf{0}$ and that $Q(\widehat{\boldsymbol{\theta}})$ is close to a maximum.

Stata has three main stopping criteria: these are the small change in the coefficient vector (`tolerance()`); the small change in the objective function (`ltolerance()`); and the small gradient relative to the Hessian (`nrtolerance()`). The Stata default values for these criteria can be changed; see `help maximize`.

The default and preferred stopping criterion is `nrtolerance()`, which is based on $\mathbf{g}(\hat{\boldsymbol{\theta}})' \mathbf{H}(\hat{\boldsymbol{\theta}})^{-1} \mathbf{g}(\hat{\boldsymbol{\theta}})$. The default is to stop when `nrtolerance()` $< 10^{-5}$.

In addition, the user should be aware that even if the iterative method has not converged, estimation will stop after `maxiter` iterations. If the maximum is reached without convergence, regression results, including parameters and standard errors, are still provided, along with a warning message that convergence is not achieved.

16.3.5 Nonconvergence and possible lack of identification

Parameter estimates should not be used if iterations fail to converge. They also should not be used if the final iteration has a warning that the objective function is nonconcave or that the Hessian is not negative definite, because this indicates that the model is not identified. Missing standard errors also indicate a problem.

Failure of iterative methods to converge may indicate numerical problems, due to poor starting values or a highly nonlinear model that is challenging to fit, that are potentially solvable. For example, convergence of estimators for the parametric models presented in chapter [23](#) might require that the number of numerical integration points be increased from program defaults.

But nonconvergence may also indicate that the model is not identified. With more complex models, fitting a model that is not identified becomes more likely. An example is if too many components are specified for a finite mixture model.

16.3.6 Multiple maximums

Complicated objective functions can have multiple optimums. The following provides an example:

```

. * Objective function with multiple optima
. graph twoway function
>     y=100-0.0000001*(x-10)*(x-30)*(x-50)*(x-50)*(x-70)*(x-80),
>     range (5 90) plotregion(style(none)) scale(1.2)
>     title("Objective function Q({&theta;}) as {&theta;} varies")
>     xtitle("{&theta;}", size(medlarge)) xscale(titlegap(*5))
>     ytitle("Q({&theta;})", size(medlarge)) yscale(titlegap(*5))

```

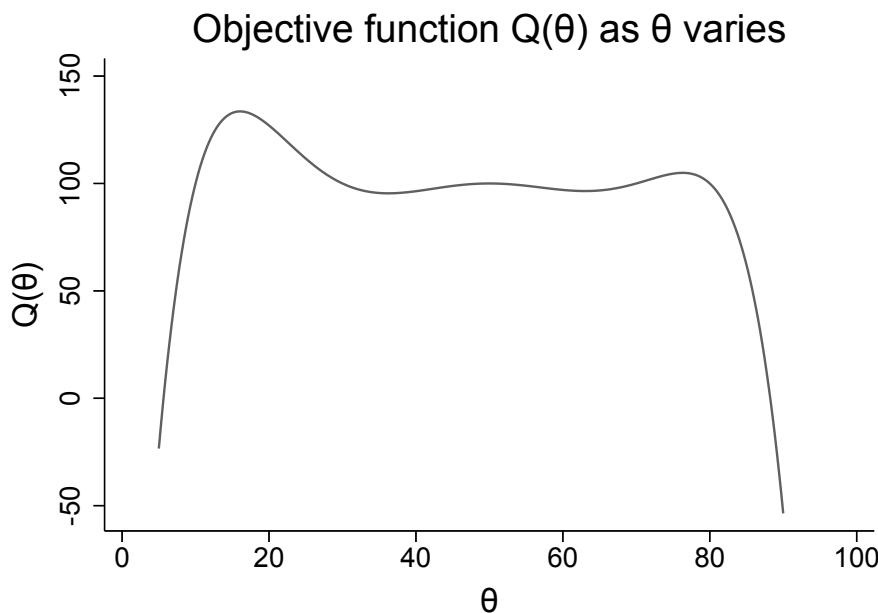


Figure 16.1. Objective function with multiple optimums

From figure 16.1, there are three local maximums—at $\theta \approx 15$, at $\theta \approx 50$, and at $\theta \approx 75$ —and two local minimums—at $\theta \approx 35$ and at $\theta \approx 65$. Most econometrics estimators are defined as a local maximum. It is assumed that the maximum is unique or that the optimizer is searching in the neighborhood of the largest of the local maximums. In this example, this approach applies to the largest of the local maximums, which is $\theta \approx 15$.

What problems might a gradient method encounter? If we start at $\theta < 30$, we will eventually move to the desired optimum at $\theta \approx 15$. If instead we start at $\theta > 30$, then we will move to smaller local maximums at $\theta = 50$ or $\theta \approx 75$. Furthermore, the objective function is relatively flat for $30 < \theta < 80$, so it may take quite some time to move to a local maximum.

Even if one obtains parameter estimates, they need not provide the largest local maximum. One method to check for multiple optima is to use a range of starting values. This problem is more likely with community-contributed estimators because most official Stata commands apply to models where multiple optima do not arise.

16.3.7 Numerical derivatives

All gradient methods require first derivatives of the objective function, and most require second derivatives. For the $q \times 1$ vector θ , there are q first derivatives and $q(q + 1)/2$ unique second derivatives that need to be calculated for each observation at each round of the iterative process, so a key component is fast computation of derivatives.

The derivatives can be computed analytically or numerically. Numerical derivatives have the attraction of simplicity but can lead to increased computation time compared with analytical derivatives. For the Poisson example in section 13.2.2, it was easy to obtain and provide analytical derivatives. We now consider numerical derivatives.

The scalar derivative $df(x)/dx = \lim_{h \rightarrow 0} [f(x + h) - f(x - h)]/2h$, so one can approximate the derivative by $\{f(x + h) - f(x - h)\}/2h$ for a suitable small choice of h . Applying this to the optimization of $Q(\theta)$, where differentiation is now with respect to a vector, for the first derivative of $Q(\hat{\theta}_s)$ with respect to the j th component of the vector θ , the numerical derivative is

$$\frac{\Delta Q(\theta)}{\Delta \theta_j} \Big|_{\hat{\theta}_s} = \frac{Q(\hat{\theta}_s + h\mathbf{e}_j) - Q(\hat{\theta}_s - h\mathbf{e}_j)}{2h}, \quad j = 1, \dots, q$$

where h is small and $\mathbf{e}_j = (0 \dots 0 \ 1 \ 0 \dots 0)'$ is a column vector with unity in the j th row and 0s elsewhere. Numerical second derivatives are calculated as the numerical first derivative of the numerical or analytical first derivative. In theory, h should be very small because formally

$\partial Q(\boldsymbol{\theta})/\partial\theta_j$ equals the limit of $\Delta Q(\boldsymbol{\theta})/\Delta\theta_j$ as $h \rightarrow 0$. But in practice, too small a value of h leads to inaccuracy due to rounding error. Stata chooses $2h$ so that $f(x + h)$ and $f(x - h)$ differ in about half their digits, or roughly 8 out of 16 digits, because computations are in double precision. This computation of h each time a derivative is taken increases accuracy at the expense of considerable increase in computation time.

The number of derivatives is greatly reduced if the objective function is an index model with few indexes. In the simplest case of a single-index model, $Q(\boldsymbol{\theta}) = N^{-1} \sum_i q(y_i, \mathbf{x}'_i \boldsymbol{\theta})$ so that $\boldsymbol{\theta}$ appears only via $\mathbf{x}'_i \boldsymbol{\theta}$. Then, by the chain rule, the gradient vector is

$$\frac{\partial Q(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{N} \sum_{i=1}^N \frac{\partial q(y_i, \mathbf{x}'_i \boldsymbol{\theta})}{\partial \mathbf{x}'_i \boldsymbol{\theta}} \times \mathbf{x}_i$$

The q scalar derivatives $\partial q(y_i, \mathbf{x}'_i \boldsymbol{\theta})/\partial\theta_j$ are the same scalar derivative $\partial q(y_i, \mathbf{x}'_i \boldsymbol{\theta})/\partial \mathbf{x}'_i \boldsymbol{\theta}$ times \mathbf{x}_i . Similarly, the $q(q+1)/2$ unique second derivatives are simple multiples of a scalar second derivative.

For a multi-index model with J indexes (often $J \leq 2$), there are J first derivatives to be calculated. Because few derivatives need to be computed, computation is slowed down little when numerical derivatives are used rather than analytical derivatives if J is small.

16.3.8 Constraints on parameters

Common constraints on parameters are that a variance should be positive and a correlation parameter should be between -1 and 1 . These constraints can be accommodated by estimating $\ln \sigma$ rather than σ and estimating $\operatorname{atanh} \rho$ rather than ρ , where $\operatorname{atanh} \rho = (1/2) \ln\{(1+\rho)/(1-\rho)\}$. Then, retransform to yield estimates $\hat{\sigma}$ and $\hat{\rho}$.

16.4 Overview of `ml`, `moptimize()`, and `optimize()`

Stata has three general optimization tools for user-defined objective functions. The `ml` command is the simplest and requires only use of Stata commands. The `moptimize()` and `optimize()` functions, by contrast, are coded in Mata and can be much faster than `ml`.

The `ml` command and the `moptimize()` function are designed for single- and multiple-index models, and the `ml` command is a Stata command front-end to the Mata function `moptimize()`. The `optimize()` function is intended for quite general objective functions that need not be index functions.

Unless computational time is a major consideration, it is best to use the simplest method possible. Note that for some objective functions, one can use the simpler `mlexp` command; see section 13.3.3. This has the advantage that postestimation commands such as `margins` and `predict` are then simple to implement.

16.4.1 Evaluator types

Each one of `ml`, `moptimize()`, and `optimize()` has several ways of defining the objective function: Stata refers to these as evaluator types. These vary according to whether

1. the objective function sums over a distinct entry for each observation, so $Q(\theta) = \sum_{i=1}^N q_i(\theta)$. In this case, an evaluator function can provide individual observations $q_i(\theta)$.
2. the parameters enter as indexes (single or multiple).
3. analytical expressions for first and second derivatives are provided.

When 1 holds, the evaluator can define the components $q_i(\theta)$, enabling computation of robust standard errors. When 2 holds, computation is sped up, as explained in the preceding subsection. And 3 also speeds up computation. For example, we use `1f0` if no derivatives are provided, `1f1` if first derivatives are provided, and `1f2` if first and second derivatives are provided.

The `lf`, `lf0`, `lf1`, and `lf2` methods, or linear-form methods, are used when both 1 and 2 hold. The most general model is then a multi-index model with J dependent variables and m indexes that specifies

$$Q(\theta) = \sum_{i=1}^N q(y_{1i}, \dots, y_{Ji}, \mathbf{x}'_{1i}\boldsymbol{\theta}_1, \dots, \mathbf{x}'_{mi}\boldsymbol{\theta}_m)$$

where Stata automatically includes a constant in each of $\mathbf{x}_{1i}, \dots, \mathbf{x}_{mi}$. For these methods, the evaluator defines each individual contribution $q_i(\boldsymbol{\theta})$, and any derivatives provided are with respect to $q_i(\boldsymbol{\theta})$.

The `d0`, `d1`, and `d2` methods, or derivative methods, are used when 2 holds but 1 does not. This is the case, for example, for estimation of panel models or clustered-data models with fixed or random effects, so that with G clusters, say, the objective function $Q(\theta)$ sums over G terms rather than N terms. Then the evaluator defines $Q(\theta)$, and any derivatives provided are with respect to $Q(\theta)$.

Robust standard errors can be directly obtained using the `lf0–lf2` methods but not using the `d0–d2` methods. The `gf0`, `gf1`, and `gf2` methods, or general-form methods, can yield robust standard errors even when 1 does not hold.

The `q0` and `q1` methods, or quadratic form methods, are for the special case where the objective function is a quadratic form $Q(\boldsymbol{\theta}) = \mathbf{h}(\boldsymbol{\theta})'\mathbf{W}\mathbf{h}(\boldsymbol{\theta})$ as used in NLS and GMM estimation.

Table 16.1 provides a summary of the various evaluators and the optimization tools for which they are available.

Table 16.1. Evaluator types for `ml`, `moptimize()`, and `optimize()`

Evaluator	<code>m1</code>	<code>moptimize()</code>	<code>optimize()</code>	Description
<code>lf</code>	✓	✓		linear form
<code>lf0</code>	✓	✓		linear form
<code>lf1</code>	✓	✓		<code>lf0</code> with gradient provided
<code>lf2</code>	✓	✓		<code>lf1</code> with Hessian provided
<code>d0</code>	✓	✓	✓	derivative
<code>d1</code>	✓	✓	✓	<code>d0</code> with gradient provided
<code>d2</code>	✓	✓	✓	<code>d1</code> with Hessian provided
<code>gf0</code>	✓	✓	✓	general form
<code>gf1</code>		✓	✓	<code>gf0</code> with gradient provided
<code>gf2</code>		✓	✓	<code>gf1</code> with Hessian provided
<code>q0</code>		✓		quadratic form
<code>q1</code>		✓		<code>q0</code> with derivative provided

Where it is relevant, the additional evaluator types `lf1debug`, `lf2debug`, `d1debug`, `d2debug`, `gf1debug`, `gf2debug`, and `q1debug` provide numerical checks that the relevant derivatives have been correctly provided.

The use of survey commands and computation of robust and cluster-robust standard errors are possible with `lf*`, `gf*`, and `q*` evaluators but not with `d*` evaluators.

16.4.2 Optimization techniques

Most of the gradient methods discussed in section [16.3.2](#) are available. Table [16.2](#) provides a summary.

Table 16.2. Optimization techniques for evaluator types for `m1`, `moptimize()`, and `optimize()`

Evaluator	<code>ml</code>	<code>moptimize()</code>	<code>optimize()</code>	Description
<code>nr</code>	✓	✓	✓	Modified NR
<code>dfp</code>	✓	✓	✓	Davidon–Fletcher–Powell
<code>bfgs</code>	✓	✓	✓	Broyden–Fletcher–Goldfarb–Shanno
<code>bhhh</code>	✓	✓	✓	Berndt–Hall–Hall–Hausman
<code>nm</code>		✓	✓	Nelder–Mead
<code>gn</code>		✓		Gauss–Newton (quadratic optimization)

The default is `nr`, except for the `moptimize()` evaluators `q0` and `q1`, which use `gn`. Note that for a given optimization tool, not all optimization techniques will be available for all evaluators. For example, for the `optimize()` function and type `d` evaluators, the `bhhh` and `gn` techniques are not available.

16.5 The `ml` command: If method

The Stata optimization command `ml` focuses on multi-index models to speed up the computation of derivatives. The name `ml` is somewhat misleading because the command can be applied to any *m* estimator (see the NLS example in section [16.5.5](#)), but in non-ML cases, one should always use a robust estimate of the VCE.

The `lf` method is the simplest method. It requires the formula for a single observation's contribution to the objective function. For ML estimation, this is the log density. The more advanced methods `lf0–lf2`, `d0–d2`, `lf0–lf2`, and `gf0` are deferred to section [16.7](#).

16.5.1 The `ml` commands

The key `ml` commands are the `ml model` command to define the model to be fit and the `ml maximize` command to perform the maximization.

The syntax for `ml model` is

```
ml model method programe eq1 [eq2 ...] [if] [in] [weight] [, options]
```

For example, `ml model lf lfpois (y=x1 x2)` will use the `lfpois` program to estimate the parameters of a single-index model with the dependent variable `y`, the regressors `x1` and `x2`, and an intercept.

The `lf`, `d0`, and `lf0` methods use only numerical derivatives, the `d1` and `lf1` methods use analytical first derivatives and numerical second derivatives, and the `d2` and `lf2` methods use only analytical derivatives. The user must provide the formulas for any analytical derivatives. Finally, for methods `d2` and `lf2` that provide expressions for second derivatives, the `negh` option is used if the negative Hessian rather than the Hessian is specified.

The syntax for `ml maximize` is

```
ml maximize [, options]
```

where many of the options are the maximization options covered in section [16.3.1](#).

There are several other `m1` commands. These include `m1 check` to check that the objective function is valid; `m1 search` to find better starting values; `m1 trace` to trace maximization execution; and `m1 init` to provide starting values.

16.5.2 The `lf` method

The simplest `m1` method is the `lf` method. This is intended for the special case where the objective function is an m estimator, simply a sum or average over the N observations of a subfunction $q_i(\boldsymbol{\theta})$, with parameters that enter as a single-index or a multi-index form. Then,

$$Q(\boldsymbol{\theta}) = \sum_{i=1}^N q(y_i, \mathbf{x}'_{i1}\boldsymbol{\theta}_1, \dots, \mathbf{x}'_{iJ}\boldsymbol{\theta}_J)$$

Usually, $J = 1$, in which case $q_i(\boldsymbol{\theta}) = q(y_i, \mathbf{x}'_{i1}\boldsymbol{\theta}_1)$, or $J = 2$. Most cross-sectional likelihoods and official Stata commands fall into this class, which Stata documentation refers to as meeting the linear-form restrictions.

The `lf` method requires that a program be written to give the formula for the subfunction $q_i(\boldsymbol{\theta})$. This program is subsequently called by the `m1 model lf` command.

The Stata documentation refers to the first index $\mathbf{x}'_{i1}\boldsymbol{\theta}_1$ as `theta1`, the second index $\mathbf{x}'_{i2}\boldsymbol{\theta}_2$ as `theta2`, and so on. This has the potential to cause confusion with the standard statistical terminology, where $\boldsymbol{\theta}$ is the generic notation for the parameter vector.

16.5.3 Poisson example: Single-index model

For the Poisson MLE, $Q(\boldsymbol{\beta}) = \sum_i q_i(\boldsymbol{\beta})$, where the log density

$$q_i(\boldsymbol{\beta}) = -\exp(\mathbf{x}'_i \boldsymbol{\beta}) + y_i \mathbf{x}'_i \boldsymbol{\beta} - \ln y_i!$$

is of single-index form.

We first write the program, referenced in `ml model`, that evaluates $q_i(\boldsymbol{\beta})$. This program has two arguments: `lnf`, for the evaluated log density for an individual observation, and `theta1`, and the single index $\mathbf{x}'_i \boldsymbol{\beta}$.

For all `ml` programs, the dependent variable y_i is assigned by Stata to the global macro `$ML_y1`. A second dependent variable would be assigned `$ML_y2`, and so on.

To improve program readability, we use the local macro `y` to substitute for `$ML_y1`, we define the temporary variable `mu` equal to `exp(theta1)`, and we define the temporary variable `lnyfact` equal to $\ln y!$. The program argument `lnf` stores the result $q_i(\boldsymbol{\beta})$:

```
. * Poisson ML program lfpois to be called by command ml method lf
. program lfpois
1.    version 17
2.    args lnf theta1          // theta1=x'b, lnf=ln(y)
3.    tempvar lnyfact mu
4.    local y "$ML_y1"         // Define y so program more readable
5.    generate double `lnyfact' = lnfactorial(`y')
6.    generate double `mu'     = exp(`theta1')
7.    qui replace `lnf'       = -`mu' + `y'*`theta1' - `lnyfact'
8. end
```

We could have more directly defined `lnf` as

```
`lnf' = -exp(`theta1') + $ML_y1*exp(`theta1') - lnfactorial($ML_y1)
```

The preceding code instead breaks this into pieces, which can be advantageous when `lnf` is complex. Stata computes `lnf` using double precision, so the intermediate variables should also be calculated in double precision. The `lnfactorial()` function is used rather than first computing $y!$ and then taking the natural logarithm because the latter method is not possible if y is at all large.

The essential commands are `ml model` and `ml maximize`. It is good practice to additionally use the `ml check` and `ml search` commands before

`ml maximize.`

For Poisson regression of `docvis` on several regressors, with a heteroskedastic-robust estimate of the VCE, we have

```
. * Command ml model, including defining y and x, plus ml check  
. ml model lf lfpois (docvis = private chronic female income), vce(robust)  
. ml check  
Test 1: Calling lfpois to check if it computes log pseudolikelihood and  
does not alter coefficient vector...  
Passed.  
Test 2: Calling lfpois again to check if the same log pseudolikelihood value  
is returned...  
Passed.  
(output omitted)
```

We then type `ml search` to try to obtain better starting values.

```
. * Search for better starting values  
. ml search  
initial:      log pseudolikelihood = -32853.851  
rescale:      log pseudolikelihood = -23375.641
```

ML estimation then occurs by typing

```

. * Command lfpois implemented for Poisson MLE
. ml maximize

initial:      log pseudolikelihood = -23375.641
rescale:      log pseudolikelihood = -23375.641
Iteration 0:  log pseudolikelihood = -23375.641
Iteration 1:  log pseudolikelihood = -21094.397
Iteration 2:  log pseudolikelihood = -18511.276
Iteration 3:  log pseudolikelihood = -18503.552
Iteration 4:  log pseudolikelihood = -18503.549
Iteration 5:  log pseudolikelihood = -18503.549

Number of obs = 4,412
Wald chi2(4) = 594.72
Prob > chi2 = 0.0000
Log pseudolikelihood = -18503.549

```

docvis	Robust					[95% conf. interval]
	Coefficient	std. err.	z	P> z		
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

Note that an intercept was automatically added. The results are the same as those given in section 13.3.2 using `poisson` and as those obtained in section [16.2.3](#) using Mata.

The following command yields cluster-robust standard errors, with clustering on `age`.

```

. * Same model but with cluster-robust standard errors
. ml model lf lfpois (docvis = private chronic female income), vce(cluster age)
. ml maximize
(output omitted)

```

16.5.4 Negative binomial example: Two-index model

A richer model for counts is the negative binomial. The log density for this model, which is explained in section [20.2.2](#), is

$$\begin{aligned}
q_i(\boldsymbol{\beta}, \alpha) = & \ln \Gamma(y_i + \alpha^{-1}) - \ln \Gamma(\alpha^{-1}) - \ln y_i! \\
& - (y_i + \alpha^{-1}) \ln \{1 + \alpha \exp(\mathbf{x}'_i \boldsymbol{\beta})\} + y_i \ln \alpha + y_i \mathbf{x}'_i \boldsymbol{\beta}
\end{aligned}$$

This introduces an additional parameter, α , so that the model is now a two-index model, with indexes $\mathbf{x}_i'\beta$ and α .

The following program computes $q_i(\beta, \alpha)$ for the negative binomial, where the two indexes are referred to as `theta1` (equals $\mathbf{x}_i'\beta$) and `a` (equals α).

```
. * Negbin ML two-index program lfnb to be called by command ml method lf
. program lfnb
1.    version 17
2.    args lnf theta1 a           // theta1=x`b, a=alpha, lnf=ln(y)
3.    tempvar mu
4.    local y $ML_y1           // Define y so program more readable
5.    generate double `mu' = exp(`theta1')
6.    qui replace `lnf' = lngamma(`y'+(1/`a')) - lngamma((1/`a'))
>          - lnfactorial(`y') - (`y'+(1/`a'))*ln(1+`a'*`mu')
>          + `y'*ln(`a') + `y'*ln(`mu')
7. end
```

The program has an additional argument, `a`, so the call to the program using `ml model` includes an additional argument `()` indicating that `a` is a constant that does not depend on regressors, unlike `theta1`. We have

```
. * Command lf implemented for negative binomial MLE
. ml model lf lfnb (docvis = private chronic female income) (), vce(robust)
. ml maximize, nolog
initial:      log pseudolikelihood =      -<inf>  (could not be evaluated)
feasible:     log pseudolikelihood = -14722.779
rescale:      log pseudolikelihood = -10743.548
rescale eq:   log pseudolikelihood = -10570.445
                                         Number of obs =  4,412
                                         Wald chi2(4) =  384.30
                                         Prob > chi2 =  0.0000
Log pseudolikelihood = -9855.1389
```

docvis	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
eq1	private	.8876559	.1284168	6.91	0.000	.6359636 1.139348
	chronic	1.143545	.0614412	18.61	0.000	1.023122 1.263967
	female	.5613027	.0663056	8.47	0.000	.4313461 .6912593
	income	.0045785	.0010666	4.29	0.000	.002488 .006669
	_cons	-.4062135	.1493729	-2.72	0.007	-.698979 -.1134479
eq2	_cons	1.726868	.0796025	21.69	0.000	1.57085 1.882886

The estimates are the same as those obtained by using the `nbreg` command in section 11.4.1; the standard errors differ because heteroskedastic–robust standard errors were used here.

16.5.5 NLS example: Nonlikelihood model

The preceding examples were likelihood based, but other m estimators can be considered.

In particular, consider NLS estimation with an exponential conditional mean. Then $Q_N(\beta) = 1/N \sum_{i=1}^N \{y_i - \exp(\mathbf{x}'_i \beta)\}^2$. This is easily estimated by typing

```
. * NLS program lfnls to be called by command ml method lf
. program lfnls
1.      version 17
2.      args lnf theta1          // theta1=x`b, lnf=squared residual
3.      local y "$ML_y1"        // Define y so program more readable
4.      qui replace `lnf' = -(`y'-exp(`theta1'))^2
5. end
```

Note the minus sign in the definition of `lnf` because the program is designed to maximize, rather than minimize, the objective function.

Running this program, we obtain

```
. * Command lf implemented for NLS estimator
. ml model lf lfnls (docvis = private chronic female income), vce(robust)
. ml maximize
(output omitted)
```

The results, omitted here, give the same coefficient estimates as those obtained from the `n1` command, which are given in section 13.3.6. The corresponding robust standard errors differ, however, by as much as 5%. The reason is that `n1` uses the expected Hessian in forming the robust estimate of the VCE (see section 13.4.5), exploiting additional information about the NLS estimator. The `ml` method instead uses the empirical Hessian. For NLS, these two differ, whereas for Poisson MLE, they do not.

For this example, the default estimate of the VCE, the inverse of the negative Hessian matrix, will always be wrong. To see this, consider

ordinary least squares (OLS) in the linear model. Then

$Q_N(\beta) = (\mathbf{y} - \mathbf{X}\beta)'(\mathbf{y} - \mathbf{X}\beta)$ has a Hessian of $-2 \times \mathbf{X}'\mathbf{X}$. Even if the errors are homoskedastic, `m1` would give an estimate of $(1/2)(\mathbf{X}'\mathbf{X})^{-1}$ rather than $s^2(\mathbf{X}'\mathbf{X})^{-1}$. Whenever `m1` is used to optimize models that are not likelihood based, a robust estimate of the VCE must be used.

16.6 Checking the program

The initial challenge is to debug a program and get it to successfully run, meaning that iterations converge and plausible regression output is obtained. There is a great art to this, and there is no replacement for experience. There are many ways to make errors, especially given the complexity of program syntax.

The next challenge is to ensure that computations are done correctly to verify that plausible output is indeed correct output. This is feasible if one can generate simulated data that satisfy model assumptions.

We focus on community-contributed programs for `ml`, but many of the following points apply to evaluating any estimator.

16.6.1 Program debugging using `ml check` and `ml trace`

The `ml check` command provides a check of the code to ensure that one can evaluate `lnf`, though this does not ensure that the evaluation is correct.

This command is most useful for checking program syntax because it provides much more detailed information than if we instead proceed directly to `ml maximize`.

For example, suppose in the `lfpois` program we typed the line

```
. generate double `mu' = exp(`theta1')
```

The mistake is that `'theta1'` was typed rather than `'theta1'`. The `ml maximize` command leads to failure and the following error message:

```
invalid syntax  
r(198);
```

This message is not particularly helpful. If instead we type

```
. ml search
```

before `ml maximize`, the program again fails, but the output now includes

```
- generate double `mu' = exp(`heta1')
= generate double __000006 = exp(
invalid syntax
```

which indicates that the error is due to a problem with ‘`heta1`’.

More complete information is given by the `ml trace` command. If we type

```
. ml trace on
```

before `ml maximize`, the program fails, and we get essentially the same output as when using `ml search`. Once the program is corrected and runs successfully, `ml trace` provides extensive details on the execution of the program. In this example, 980 lines of detail are given.

The trace facility can also be used for commands other than `ml` by typing

```
. set trace on
```

A drawback of using `trace` is that it can produce copious output.

A more manual-targeted method to determine where problems may arise in a program is to include messages in the program. For example, suppose we place in the `lfpois` program the line

```
display "I made it to here"
```

If the program fails after this line is displayed, then we know that the problem arose beyond the line where the `display` statement was given.

16.6.2 Getting the program to run

The `ml check` command essentially checks program syntax. This does not protect against other coding errors such as misspecification of the log density. Suppose, for example, that in the `lfpois` program we typed

```
quietly replace `lnf' = `mu' + `y'*ln(`mu') - `lnyfact'
```

The error here is that we have ‘`mu`’ rather than `-‘mu’`. Then we obtain

```
. ml maximize
initial:      log pseudolikelihood = -25075.609
alternative:   log pseudolikelihood = -13483.451
(4412 missing values generated)
rescale:       log pseudolikelihood =  1.01e+226
Iteration 0:   log pseudolikelihood =  1.01e+226 (not concave)
(1 missing value generated)
Iteration 1:   log pseudolikelihood =  1.76e+266 (not concave)
(762 missing values generated)
Hessian has become unstable or asymmetric (NC)
r(504);
```

Here the error has occurred quite early. One possibility is that poor starting values were given. But using `ml search` leads to far worse starting values in this example. In this case, the most likely explanation is an error in the objective function.

Poor starting values can lead to problems if the objective function is not globally concave. For index models, a good approach is to set all parameters to zero aside from the constant, which is set to that value appropriate for an intercept-only model. For example, for the Poisson intercept-only model with the parameter α , we have $\hat{\alpha} = \ln \bar{y}$ because then $\exp(\hat{\alpha}) = \bar{y}$. Thus, the initial value is $(0, \dots, \ln \bar{y})$. It can be useful to try the model with few regressors, such as with an intercept and a single regressor.

The `ml` methods `d1`, `d2`, `lf1`, and `lf2`, presented in section [16.7](#), require analytical expressions for the first and second derivatives. The `ml model d1debug` and `ml model d2debug` commands check these by comparing these expressions with numerical first and second derivatives. Any substantial difference indicates error in coding the derivatives or, possibly, the original objective function. There are similar methods `lf1debug` and `lf2debug`.

16.6.3 Checking the data

A common reason for program failure is that the program is fine but that the data passed to the program are not.

For example, the `lfpois` program includes the `lnfactorial('y')` function, which requires that 'y' be a nonnegative integer. Consider the impact of the following:

```
. replace docvis = 0.5 if docvis == 1  
. ml model lf lfpois (docvis = private chronic female income), vce(robust)  
. ml maximize
```

The resulting output after `ml maximize` includes many lines with

(700 missing values generated)

followed by

```
could not find feasible values  
r(491);
```

One should always use `summarize` to obtain summary statistics of the dependent variable and regressors ahead of estimation as a check. Indications of problems include an unexpected range, zero standard deviation, and missing values. In this particular example, `summarize` will not detect the problem, but `tabulate docvis` will.

16.6.4 Multicollinearity and near collinearity

If variables are perfectly collinear, then Stata estimation commands, including `ml`, detect multicollinearity and drop the regressors as needed.

If variables are close to perfectly collinear, then numerical instability may cause problems. We illustrate this by adding the additional regressor `extra`, which equals `income` plus au , where u is a draw from the uniform distribution and $a = 0.001$ or 0.01 .

First, add the regressor `extra`, which equals `income` plus $0.001u$. We have

```
. * Example with high collinearity interpreted as perfect collinearity  
. generate extra = income + 0.001*runiform()  
. ml model lf lfpois (docvis = private chronic female income extra), vce(robust)  
note: extra omitted because of collinearity.
```

Here `ml maximize` interprets this as perfect collinearity and drops `income` before maximization.

Next, instead add the regressor `extra2`, which equals `income` plus the larger amount $0.01u$. We have

```

. * Example with high collinearity not interpreted as perfect collinearity
. generate extra2 = income + 0.01*runiform()
. ml model lf lfpois (docvis = private chronic female income extra2), vce(robust)
. ml maximize, nolog
                                         Number of obs = 4,412
                                         Wald chi2(5) = 593.92
                                         Prob > chi2 = 0.0000
Log pseudolikelihood = -18501.885

```

docvis	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
private	.7982791	.1091528	7.31	0.000	.5843436 1.012215
chronic	1.092004	.0559761	19.51	0.000	.9822934 1.201715
female	.4922122	.0586045	8.40	0.000	.3773495 .607075
income	-4.75283	10.0911	-0.47	0.638	-24.53102 15.02536
extra2	4.7564	10.09136	0.47	0.637	-15.0223 24.5351
_cons	-.2535819	.1154495	-2.20	0.028	-.4798587 -.0273051

Now this is no longer interpreted as perfect collinearity, so estimation proceeds. The coefficients of `income` and `extra2` are very imprecisely estimated, while the remaining coefficients and standard errors are close to those given in section 13.3.2.

Pairwise collinearity can be detected by using the `correlate` command, and multicollinearity can be detected with the `_rmcoll` command. For example,

```

. * Detect multicollinearity by using _rmcoll
. _rmcoll income extra
note: extra omitted because of collinearity.
. _rmcoll income extra2

```

Another simple data check is to see whether the parameters of the model can be estimated by using a closely related Stata estimation command. For the `lfpois` program, the obvious test is regression using `poisson`. If a command to compute Poisson regression was not available, we could at least try OLS regression, using `regress`.

16.6.5 Checking parameter estimation

Once a program runs, we need to check that it is correct. The following approach is applicable to estimation with any method, not just with the `ml` command.

To check parameter estimation, we generate data from the same data-generating process (DGP) as that justifying the estimator for a large sample size N . Because the desired estimator is consistent as $N \rightarrow \infty$, we expect the estimated parameters to be very close to those of the DGP. A similar exercise was done in section 5.6.

To check a Poisson model-estimation program, we generate data from the following DGP:

$$y_i = \text{Poisson}(\beta_1 + \beta_2 x); \quad (\beta_1, \beta_2) = (2, 1); \quad x_i \sim N(0, 0.5); \quad i = 1, \dots, 10000$$

The following code generates data from this DGP:

```
. * Generate dataset from Poisson DGP for large N
. clear
. set obs 10000
Number of observations (_N) was 0, now 10,000.
. set seed 10101
. generate x = rnormal(0,0.5)
. generate mu = exp(2 + x)
. generate y = rpoisson(mu)
. summarize mu x y
```

Variable	Obs	Mean	Std. dev.	Min	Max
mu	10,000	8.384263	4.430457	.9421767	44.62779
x	10,000	.0006259	.5038249	-2.059562	1.798357
y	10,000	8.3369	5.271165	0	53

The normal regressor has the desired mean and variance, and the count outcome has a mean of 8.34 and ranges from 0 to 53.

We then run the previously defined `lfpois` program. Because the DGP is known to be Poisson here, we simply use default standard errors.

. * Consistency check: Run program lfpois and compare beta to DGP value					
. ml model lf lfpois (y = x)					
. ml maximize, nolog					
Log likelihood = -23995.411	Number of obs = 10,000 Wald chi2(1) = 20456.00 Prob > chi2 = 0.0000				
	<hr/>				
y	Coefficient	Std. err.	z	P> z	[95% conf. interval]
x	.9978772	.006977	143.02	0.000	.9842026 1.011552
_cons	1.994867	.0038788	514.30	0.000	1.987265 2.002469

The estimates are $\hat{\beta}_2 = 0.998$ and $\hat{\beta}_1 = 1.995$, quite close to the DGP values of 1 and 2. We expect the 95% confidence interval for $\hat{\beta}_2$ to include the DGP value of 1 in 95% of such simulations. The 95% confidence interval of this simulation is [0.984, 1.012], which includes 1.

If $N = 1000000$, for example, estimation is more precise, and we expect estimates to be very close to the DGP values.

This DGP is quite simple. More challenging tests would consider a DGP with additional regressors from other distributions.

16.6.6 Checking standard error estimation

To check that standard errors for an estimator $\hat{\beta}$ are computed correctly, we can perform, say, $S = 2000$ simulations that yield S estimates $\hat{\beta}$ and S computed standard errors $s_{\hat{\beta}}$. If the standard errors are correctly estimated, then the average of the S computed standard errors, $\bar{s}_{\hat{\beta}} = S^{-1} \sum_{s=1}^S s_{\hat{\beta}}$, should equal the standard deviation of the S estimates $\hat{\beta}$, which is $(S-1)^{-1} \sum_{s=1}^S (\hat{\beta}_s - \bar{\hat{\beta}})^2$, where $\bar{\hat{\beta}} = S^{-1} \sum_{s=1}^S \hat{\beta}_s$. The sample size needs to be large enough that we believe that asymptotic theory provides a good guide for computing the standard errors. We set $N = 500$.

We first write the `secheck` program, which draws one sample from the same DGP as was used in the previous section. Again, default standard errors are used.

```

. * Program to generate dataset, obtain estimate, and return beta and SEs
. program secheck, rclass
1.      version 17
2.      drop _all
3.      set obs 500
4.      generate x = rnormal(0,0.5)
5.      generate mu = exp(2 + x)
6.      generate y = rpoisson(mu)
7.      ml model lf lfpois (y = x)
8.      ml maximize
9.      return scalar b1 = _b[_cons]
10.     return scalar se1 = _se[_cons]
11.     return scalar b2 = _b[x]
12.     return scalar se2 = _se[x]
13. end

```

We then run this program 2,000 times, using the `simulate` command. (The `postfile` command could alternatively be used.) We have

```

. * Standard errors check: Run program secheck
. set seed 10101
. simulate "secheck" bcons=r(b1) se_bcons=r(se1) bx=r(b2) se_bx=r(se2), reps(2000)
Command:      secheck
Statistics:   bcons      = r(b1)
              se_bcons   = r(se1)
              bx         = r(b2)
              se_bx     = r(se2)

. summarize

```

Variable	Obs	Mean	Std. dev.	Min	Max
bcons	2,000	2.000752	.0170348	1.930763	2.061243
se_bcons	2,000	.0172855	.0002088	.0166568	.0180361
bx	2,000	.9998756	.0310063	.9022716	1.113206
se_bx	2,000	.0310439	.0015591	.0255744	.0361679

The column `Obs` in the summary statistics here refers to the number of simulations ($S = 2000$). The actual sample size, set inside the `secheck` program, is $N = 500$.

For the intercept, we have $\overline{s_{\hat{\beta}_1}} = 0.0173$ compared with 0.0170 for the standard deviation of the 2,000 estimates for $\hat{\beta}_1$ (`bcons`). For the slope, we have $\overline{s_{\hat{\beta}_2}} = 0.0310$ compared with 0.0310 for the standard deviation of the 2,000 estimates for $\hat{\beta}_2$ (`bx`). The standard errors are correctly estimated.

16.7 The `ml` command: `lf0–lf2`, `d0–d2`, and `gf0` methods

In the following discussion, the terms “log density” and “log likelihood” are used, but more generally, the methods apply to m estimators. Then the log-likelihood function is the objective function, and log density is the contribution of a single observation. In the nonlikelihood case, however, robust standard errors should be used.

The `lf` method is fast and simple when the objective function is of the form given in section [16.5.2](#), a form that Stata manuals refer to as the linear form; then parameters appear as a single index. The `lf` method does not require specification of derivatives.

The `lf0`, `lf1`, and `lf2` methods are like method `lf` in that they are used when the objective function meets the linear-form restrictions and is a sum of individual terms for each observation. These methods require you to write evaluator programs that are more complicated than those for method `lf`, in exchange for the ability to specify first and second derivatives of the log density with methods `lf1` and `lf2`. This can speed up computation compared with the `lf` and `lf0` methods that rely on numerical derivatives. If you do not plan to provide derivatives, there is no point in using method `lf0`, because method `lf` provides the same functionality and is easier to use.

The `d0`, `d1`, and `d2` methods are more general than `lf`. Again, parameters enter through indexes, so the linear form restrictions are used, but now one can accommodate situations where there are multiple observations or equations for each individual in the sample. This can arise with fixed- and random-effects models for panel data and clustered data, with conditional multinomial logit models, where regressor values vary over each of the potential outcomes, in systems of equations, and in the Cox proportional hazards model, where a risk set is formed at each failure. The evaluator program for method `d0` defines the log likelihood, whereas method `d1` allows one to provide analytical expressions for the gradient, and method `d2` additionally allows an analytical expression for the Hessian.

The evaluators for the `lf*` methods provide the log density and, where relevant, its derivatives, whereas the `d*` evaluators provide the objective

function and, where relevant, its derivatives. In both cases, the objective functions and their derivatives are with respect to the indexes $\mathbf{x}'_{ji}\theta_j$ rather than θ_j .

16.7.1 ml evaluator functions

For the `d0-d2` and `lf0-lf2` methods, the syntax is

```
ml model method progrname eq1 [eq2 ...] [if] [in] [weight] [, options]
```

where *method* is `d0`, `d1`, `d2`, `lf0`, `lf1`, or `lf2`; *progrname* is the name of an evaluator program; *eq1* defines the dependent variables and the regressors involved in the first index; *eq2* defines the regressors involved in the second index; and so on.

The evaluator program, *progrname*, has five arguments for `d0`, `d1`, and `d2` evaluators: `todo`, `b`, `lnf`, `g`, and `H`. The `ml` command uses the `todo` argument to request no derivative, the gradient, or the gradient and the Hessian. The `b` argument is the row vector of parameters θ . The `lnf` argument is the scalar objective function $Q(\theta)$. The `g` argument is a row vector for the gradient $\partial Q(\theta)/\partial\theta'$, which needs to be provided only for the `d1` and `d2` methods. The `H` argument is the Hessian matrix $\partial^2 Q(\theta)/\partial\theta\partial\theta'$, which needs to be provided for the `d2` method.

The evaluators for the `d0-d2` methods first need to link the parameters θ to the indexes $\mathbf{x}'_{1i}\theta_1, \dots$. This is done with the `mleval` command, which has the syntax

```
mleval newvar = vecname [, eq(#)]
```

For example, `mleval 'theta1'='b', eq(1)` labels the first index $\mathbf{x}'_{1i}\theta_1$ as `theta1`. The variables in \mathbf{x}_{1i} will be listed in *eq1* in `ml model`.

Next, the evaluator needs to compute the objective function $Q(\theta)$, unlike the `lf` method, where the *i*th entry $q_i(\theta)$ in the objective function is computed. The `mlsum` command sums $q_i(\theta)$ to yield $Q(\theta)$. The syntax is

```
mlsum scalarname_lnf = exp [if] [, noweight]
```

For example, `m1sum 'lnf'=(`y'-`theta1')^2` computes the sum of squared residuals $\sum_i(y_i - \mathbf{x}'_{1i}\boldsymbol{\theta}_1)^2$.

The `d1` and `d2` methods require specification of the gradient of the overall log likelihood $Q(\boldsymbol{\theta})$. For linear-form models, this computation can be simplified with the `mlvecsum` command, which has the syntax

```
mlvecsum scalarname_lnf rowvecname = exp [if] [, eq(#) ]
```

For example, `mlvecsum 'lnf' 'd1'='y'-`theta1'` computes the gradient for the subset of parameters that appears in the first index as the row vector $\sum_i(y_i - \mathbf{x}'_{1i}\boldsymbol{\theta}_1)\mathbf{x}_{1i}$. Note that `mlvecsum` automatically multiplies `'y'-`theta1'` by the regressors \mathbf{x}_{1i} in the index `theta1` because equation one is the default when `eq()` is not specified.

The `1f0–1f2` methods, like the `d0–d2` methods, require you to first use `mleval` to obtain the indexes $\mathbf{x}'_{1i}\boldsymbol{\theta}_1, \dots$. Method `1f0` evaluators receive three arguments: `todo`, `b`, and `lnfj`. The variable `lnfj` is to be filled in with the observation-level log-likelihood $q_i(\boldsymbol{\theta})$. The variable is named `lnfj`, rather than `lnfi`, because official Stata documentation for the `m1` command uses j , rather than i , to denote the typical observation. The key difference between methods `1f` and `1f0` is that the former passes to *progname* the indexes as `theta1`, `theta2`, etc., while the latter passes to your program the parameter vector `b` from which you must obtain the indexes yourself.

The `1f1` and `1f2` methods require specification of the observation-level scores associated with the log-likelihood function. That is, the derivatives of the log-likelihood function with respect to the indexes $\mathbf{x}'_{1i}\boldsymbol{\theta}_1, \mathbf{x}'_{2i}\boldsymbol{\theta}_2, \dots$. Whereas you specify the derivatives $\partial Q(\boldsymbol{\theta})/\partial\boldsymbol{\theta}_j$ with methods `d1` and `d2`, methods `1f1` and `1f2` require you to fill in variables containing $\partial q_i(\boldsymbol{\theta})/\partial\mathbf{x}'_{ji}\boldsymbol{\theta}_j$. The predominant advantage of these methods is speed: evaluating analytic derivatives is much faster than computing them numerically. Moreover, with observation-level first derivatives, `m1` can compute the robust estimate of the VCE, which is not possible with methods `d0–d2`.

With method `1f1`, *progname* receives $3 + J$ arguments, where J is the number of indexes. For a single-index model, *progname* will receive four

arguments: `todo`, `b`, `lnfj`, and `g1`. When `todo==1`, in addition to filling in `lnfj` with the observation-level log likelihood, you are to fill in the variable `g1` with $\partial q_i(\theta)/\partial \mathbf{x}'_{ji} \theta_j$. Method `lf2` receives $4 + J$ arguments; the final argument, `H`, is a matrix to be filled in with the Hessian of the overall log-likelihood function when `todo==2`.

The `d2` and `lf2` methods require specification of the Hessian matrix, the default, or the negative Hessian matrix, in which case the `negh` option is added to the `ml model` command. For linear-form models, this computation can be simplified with the `mlmatsum` command, which has the syntax

```
mlmatsum scalarname_lnf matrixname = exp [if] [, eq(#, #)]
```

For example, `mlmatsum 'lnf' 'd1'='theta1'` computes the negative Hessian matrix for the subset of parameters that appears in the first index as $\sum_i \mathbf{x}'_{1i} \theta_1$. The `mlmatsum` command automatically multiplies '`theta1`' by $\mathbf{x}_{1i}\mathbf{x}'_{1i}$, the outer product of the regressors \mathbf{x}_{1i} in the index `theta1`.

16.7.2 ml methods lf0, lf1 and lf2

We consider the cross-sectional Poisson model, a single-index model. For multi-index models such as the Weibull and panel data, Cox proportional hazards, and conditional logit models, see the Weibull example in [R] **ml** and [Gould, Pitblado, and Poi \(2010\)](#), who also consider complications such as how to make ado-files and how to incorporate sample weights.

An `lf0` method evaluator program for the Poisson MLE is the following:

```
. * ml method lf0: Program lf0pois gives lnf(y_i) in terms of index x_i`b
. program lf0pois
1.      version 17
2.      args todo b llnfi
3.      tempvar theta1          // theta1 = x`b where x given in eq(1)
4.      mleval `theta1' = `b', eq(1)
5.      local y $ML_y1          // Define y so program more readable
6.      qui replace `lnf1' = -exp(`theta1') + `y'*`theta1' - lnfactorial(`y')
7. end
```

The code is similar to that given earlier for the `lf` method. We add the additional argument `todo`, which is not used in an `lf0` program but is used in

`lf1` and `lf2` programs. And we add `mleval` to form $\mathbf{x}'\boldsymbol{\beta}$. Here there is one dependent variable and only one index.

We specify the dependent variable to be `docvis` and the regressors to be `private`, `chronic`, `female`, and `income`, plus an intercept. The following code requests heteroskedastic-robust standard errors.

```
. qui use mus210mepsdocvisyoung, clear
. qui keep if year02 == 1
. generate cons = 1
. * ml method lf0: Obtain Poisson MLE with heteroskedastic-robust standard errors
. ml model lf0 lf0pois (docvis = private chronic female income), vce(robust)
. ml maximize

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
Number of obs = 4,412
Wald chi2(4)  = 594.72
Prob > chi2   = 0.0000
Log pseudolikelihood = -18503.549
```

docvis	Robust					[95% conf. interval]
	Coefficient	std. err.	z	P> z		
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 resulting coefficient estimates are the same as those from both the `poisson` command and from the `lf` method given in section 16.5.3. The robust estimate of the VCE is that given in section 13.4.5, with $\hat{\mathbf{H}}_i$ computed by using numerical derivatives. The standard errors are exactly the same as those obtained using the `poisson` command with the `vce(robust)` option.

The `lf1` method evaluator program additionally provides an analytical expression for the derivative of the log density of the i th observation with respect to the index $\mathbf{x}'_{1i}\boldsymbol{\theta}_1$. We add the argument `gi` to the program. We have

```

. * ml method lf1: Program lf1pois adds analytical first derivatives
. program lf1pois
1.    version 17
2.    args todo b llnfi gi
3.    tempvar theta1           // theta1 = x`b where x given in eq(1)
4.    mleval `theta1' = `b', eq(1)
5.    local y $ML_y1           // Define y so program more readable
6.    qui replace `lnfi' = -exp(`theta1') + `y'^`theta1' - lnfactorial(`y')
7.    if (`todo'==0) exit
8.    qui replace `gi' = `y' - exp(`theta1') // Extra code for robust
9. end

```

The model is run in the same way, with `lf0` replaced by `lf1` and the evaluator function `lf0pois` replaced by `lf1pois`. Applying this program to the same data, but instead obtaining cluster-robust standard errors with clustering on `age`, we obtain

```

. * ml method lf1: Implement Poisson MLE with cluster--robust standard errors
. ml model lf1 lf1pois (docvis = private chronic female income), vce(cluster age)
. ml maximize, nolog
                                         Number of obs = 4,412
                                         Wald chi2(4) = 657.72
Log pseudolikelihood = -18503.549                                         Prob > chi2 = 0.0000
                                         (Std. err. adjusted for 40 clusters in age)

```

docvis	Robust					
	Coefficient	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

The `lf2` method evaluator program additionally provides an analytical expression for the matrix of second derivatives of the log density of the i th observation with respect to the index. This adds three lines of code, using the `mlmatsum` command, that are identical to that given in the `d2` method evaluator example given in the next subsection.

16.7.3 ml methods d0, d1, and d2

A `d0` method evaluator program for the Poisson MLE is the following:

```

. * ml method d0: Program lf0pois gives lnf(y) in terms of index x_i`b
. program d0pois
1.      version 17
2.      args todo b lnf          // todo is not used, b=b, lnf=lnL
3.      tempvar theta1         // theta1=x`b given in eq(1)
4.      mleval `theta1' = `b', eq(1)
5.      local y $ML_y1          // Define y so program more readable
6.      mlsum `lnf' = -exp(`theta1') + `y'^`theta1' - lnfactorial(`y')
7. end

```

The code is similar to that given earlier for the `lf` methods. The major difference is that the program calculates the log likelihood for the sample, rather than the log density for each observation. The `mlsum` command forms the objective function as the sum of log densities for each observation.

Applying this program to the same data, we obtain

```

. * ml method d0: Poisson MLE with default standard errors only possible
. ml model d0 d0pois (docvis = private chronic female income)
. ml maximize, nolog

```

Number of obs =	4,412
Wald chi2(4) =	8052.34
Prob > chi2 =	0.0000

Log likelihood = -18503.549

docvis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
private	.7986653	.027719	28.81	0.000	.7443371 .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	-.2859815 -.173471

The resulting coefficient estimates are the same as those from the `poisson` command and those using the `lf` method given in section [16.5.3](#).

Note that only default standard errors are reported. This is a consequence of the `d0`, `d1`, and `d2` programs computing the derivative of the log likelihood, rather than the derivative of the log density for each observation. This does not provide enough information to compute the middle term in the formula for the robust estimate of the VCE that is given in section 13.4.5.

We next consider an example where analytical first and second derivatives of the log likelihood are provided. The `d2` method evaluator program must provide analytical expressions for the gradient Hessian.

```

. * ml method d2: Program d2pois adds analytical first and second derivatives
. program d2pois
1.    version 17
2.    args todo b lnf g H          // Add g and H to the arguments list
3.    tempvar theta1             // theta1 = x'b where x given in eq(1)
4.    mleval `theta1' = `b', eq(1)
5.    local y $ML_y1           // Define y so program more readable
6.    mlsum `lnf' = -exp(`theta1') + `y'^*`theta1' - lnfactorial(`y')
7.    if (`todo'==0 | `lnf'>=.) exit // d1 extra code from here
8.    tempname d1
9.    mlvecsum `lnf' `d1' = `y' - exp(`theta1')
10.   matrix `g' = (`d1')
11.   if (`todo'==1 | `lnf'>=.) exit // d2 extra code from here
12.   tempname d11
13.   mlmatsum `lnf' `d11' = exp(`theta1')
14.   matrix `H' = -`d11'
15. end

```

The `mlvecsum` command forms the gradient row vector $\sum_i \{y_i - \exp(\mathbf{x}_i'\hat{\beta})\}\mathbf{x}_i'$, where \mathbf{x}_i are the first-equation regressors. The `mlmatsum` command forms minus the Hessian matrix $\sum_i \exp(\mathbf{x}_i'\hat{\beta})\mathbf{x}_i\mathbf{x}_i'$, where \mathbf{x}_i are the first-equation regressors.

We obtain

```

. * ml method d2: Implement Poisson MLE with default standard errors only possible
. ml model d2 d2pois (docvis = private chronic female income)
. ml maximize, nolog

```

	Number of obs = 4,412
	Wald chi2(4) = 8052.34
	Prob > chi2 = 0.0000
Log likelihood = -18503.549	

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

Again, only default standard errors can be obtained.

With more than one index, it will be necessary to compute cross-derivatives such as '`d12`'. The `mlmatbysum` command is an extension that can be applied when the log likelihood for the i th observation involves a

grouped sum, such as for panel data. See [R] **ml** for a two-index example, the Weibull MLE.

16.7.4 ml method gf0

The `gf*` methods, like the `lf*` methods, provide the log-likelihood contribution of each observation. The `gf0` program is

```
. * ml method gf0: Program gf0pois gives lnf(y) in terms of index x_i`b
. program gf0pois
1.      version 17
2.      args todo b lnf          // todo is not used, b=b, lnf=lnL
3.      tempvar theta1         // theta1=x`b given in eq(1)
4.      mleval `theta1' = `b', eq(1)
5.      local y $ML_y1          // Define y so program more readable
6.      qui replace `lnf' = -exp(`theta1') + `y'*`theta1' - lnfactorial(`y')
7. end
```

And we obtain

```
. * ml method gf0: Implement Poisson MLE with heteroskedastic-robust standard
> errors
. ml model gf0 gf0pois (docvis = private chronic female income), vce(robust)
. ml maximize, nolog
```

Log pseudolikelihood = -18503.549	Number of obs = 4,412 Wald chi2(4) = 594.72 Prob > chi2 = 0.0000
-----------------------------------	--

docvis	Coefficient	Robust		z	P> z	[95% conf. interval]
		std. err.				
private	.7986653	.1090015	7.33	0.000	.5850263	1.012304
chronic	1.091865	.0559951	19.50	0.000	.9821166	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	-.0124186

16.8 Nonlinear instrumental-variables (GMM) example

As an example of a nonlinear estimation problem that cannot be solved using the `m1` command, we consider GMM estimation, or more specifically nonlinear instrumental-variables (IV) estimation, of a Poisson model with endogenous regressors.

The two-stage least-squares interpretation of linear IV does not extend to nonlinear models, so we cannot simply do Poisson regression with the endogenous regressor replaced by fitted values from a first-stage regression. There are several possible methods to control for endogeneity; see section [20.7](#). We consider use of the nonlinear IV estimator, introduced in section 13.3.9, where estimation used the `gmm` command. The model can also be fit using the `ivpoisson gmm` command; see section [20.7.3](#).

Here we instead illustrate one of the more general optimization commands. The objective function is a quadratic form in sums rather than a simple sum, so it is not suited for the Stata `m1` command. So we use the Mata `optimize()` function.

16.8.1 Nonlinear IV example

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

$$E [\mathbf{z}_i \{y_i - \exp(\mathbf{x}'\boldsymbol{\beta})\}] = \mathbf{0}$$

Then the nonlinear IV estimator, a GMM estimator, minimizes

$$Q(\boldsymbol{\beta}) = \mathbf{h}(\boldsymbol{\beta})' \mathbf{W} \mathbf{h}(\boldsymbol{\beta}) \tag{16.4}$$

where the $r \times 1$ vector $\mathbf{h}(\boldsymbol{\beta}) = \sum_i \mathbf{z}_i \{y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})\}$. This is a special case of (13.5) with $\mathbf{h}(\mathbf{w}_i, \boldsymbol{\theta}) = \mathbf{z}_i \{y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})\}$.

Define the $r \times K$ matrix $\mathbf{G}(\boldsymbol{\beta}) = -\sum_i \exp(\mathbf{x}'_i \boldsymbol{\beta}) \mathbf{z}_i \mathbf{x}'_i$. Then the $K \times 1$ gradient vector

$$\mathbf{g}(\boldsymbol{\beta}) = \mathbf{G}(\boldsymbol{\beta})' \mathbf{W} \mathbf{h}(\boldsymbol{\beta}) \quad (16.5)$$

and the $K \times K$ expected Hessian is

$$\mathbf{H}(\boldsymbol{\beta}) = \mathbf{G}(\boldsymbol{\beta})' \mathbf{W} \mathbf{G}(\boldsymbol{\beta})'$$

where simplification has occurred by using $E\{\mathbf{h}(\boldsymbol{\beta})\} = \mathbf{0}$.

The estimate of the VCE is that in (13.6) with $\widehat{\mathbf{G}} = \mathbf{G}(\widehat{\boldsymbol{\beta}})$ and $\widehat{\mathbf{S}} = \sum_i \{y_i - \exp(\mathbf{x}'_i \widehat{\boldsymbol{\beta}})\}^2 \mathbf{z}_i \mathbf{z}'_i$.

16.8.2 GMM using the Mata optimize() function

The first-order conditions $\mathbf{g}(\boldsymbol{\beta}) = \mathbf{0}$, where $\mathbf{g}(\boldsymbol{\beta})$ is given in (16.5), have no solution for $\boldsymbol{\beta}$, so we need to use an iterative method. The `m1` command is not well suited to this optimization because $Q(\boldsymbol{\beta})$ given in (16.4) is a quadratic form.

Instead, we use the Mata `optimize()` function. Necessary background information on Mata and `optimize()` is provided in appendixes B and C, and it is helpful to also see the Mata OLS example in section 3.9.

We let $\mathbf{W} = (\sum_i \mathbf{z}_i \mathbf{z}'_i)^{-1}$ as for linear two-stage least squares. The following Mata expressions form the desired quantities, where we express the parameter vector `b` and gradient vector `g` as row vectors because the `optimize()` function requires row vectors. We have

```

: Xb = X*b'                                // b for optimize is 1 x k row vector
: mu = exp(Xb)
: h = Z`*(y-mu)                            // h is r x 1 column vector
: W = cholinv(Z`Z)                          // W is r x r wmatrix
: G = -(mu:*Z)`X                           // G is r x k matrix
: S = ((y-mu):*Z)`((y-mu):*Z)             // S is r x r matrix
: Qb = h`W*h                                // Q(b) is scalar
: g = (G`W*h)`                             // Gradient for optimize is a 1 x k row vector
: H = G`W*G                                 // Hessian for optimize is k x k matrix
: V = luinv(G`W*G)*G`W*S*W*G*luinv(G`W*G)

```

We fit a model for `docvis`, where `private` is endogenous and `firmsize` is used as an instrument, so the model is just identified. We use `optimize()` method `d2`, where the objective function is given as a scalar and both a gradient vector and Hessian matrix are provided. The `optimize_result_V_robust(S)` command does not apply to d evaluators, so we need to compute the robust estimate of the VCE after optimization.

The structure of the Mata code is similar to that for the Poisson example explained in section C.2.3. We have

```
. * optimize() method d2: Evaluator and implement GMM estimator for Poisson
. mata
```

```
----- mata (type end to exit) -----
: mata clear
: void pgmmd2(todo, b, y, X, Z, Qb, g, H)
> {
>     Xb = X*b'
>     mu = exp(Xb)
>     h = Z`*(y-mu)
>     W = cholinv(cross(Z,Z))
>     Qb = h`W*h
>     if (todo == 0) return
>     G = -(mu:*Z)`X
>     g = (G`W*h)`
>     if (todo == 1) return
>     H = G`W*G
>     _makesymmetric(H)
> }
: st_view(y=., ., "$y")
: st_view(X=., ., tokens("$xlist"))
: st_view(Z=., ., tokens("$zlist"))
: S = optimize_init()
: optimize_init_which(S,"min")
: optimize_init_evaluator(S, &pgmmd2())
: optimize_init_evaluator_type(S, "d2")
: optimize_init_argument(S, 1, y)
: optimize_init_argument(S, 2, X)
: optimize_init_argument(S, 3, Z)
: optimize_init_params(S, J(1,cols(X),0))
: optimize_init_technique(S,"nr")
: b = optimize(S)
Iteration 0: f(p) = 71995.212
Iteration 1: f(p) = 9259.0408
Iteration 2: f(p) = 1186.8103
Iteration 3: f(p) = 3.4395408
Iteration 4: f(p) = .00006905
Iteration 5: f(p) = 5.672e-14
Iteration 6: f(p) = 1.447e-26
: // Compute robust estimate of VCE and SEs
: Xb = X*b'
: mu = exp(Xb)
: h = Z`*(y-mu)
: W = cholinv(cross(Z,Z))
: G = -(mu:*Z)`X
: n = rows(X)
: k = cols(X)
: Shat = ((y-mu):*Z)`((y-mu):*Z)*rows(n)/(n-k)
: Vb = luinv(G`W*G)*G`W*Shat*W*G*luinv(G`W*G)
: seb = (sqrt(diagonal(Vb)))`
: b \ seb
      1          2          3          4          5
1  1.340291853  1.072907529  .477817773  .0027832801  -.6832461817
2  .0234843641  .0011488761  .0010399796  .0000330189  .0203299038
```

```
: end
```

The results are the same as those from the `gmm` command given in the section 13.3.10 example.

More generally, we could include additional instruments, which requires changing only the local macro for ‘`zlist`’. The model becomes overidentified, and GMM estimates vary with choice of weighting matrix \mathbf{W} . The one-step GMM estimator is $\hat{\beta}$, given above. The two-step (or optimal) GMM estimator recalculates $\hat{\beta}$ by using the weighting matrix $\mathbf{W} = \hat{\mathbf{S}}^{-1}$. This is illustrated in section [20.7.3](#) with the `gmm` command.

The Mata code is easily adapted to other cases where $E\{y - m(\mathbf{x}'\beta)|\mathbf{z}\} = 0$ for the specified function $m(\cdot)$, so it can be used, for example, for logit and probit models.

16.9 Additional resources

The key Stata references are [R] **ml** and [R] **Maximize**. [Gould, Pitblado, and Poi \(2010\)](#) provide a succinct yet quite comprehensive overview of the `ml` method and Mata-based likelihood evaluators using `moptimize()` functions. [Drukker \(2016\)](#) provides many details on programming in Stata. [Baum \(2016\)](#) includes the `ml` method and an example using the Mata `optimize()` function. For Mata, see appendixes B and C.

Nonlinear optimization is covered in [Cameron and Trivedi \(2005, chap. 10\)](#), [Greene \(2018, app. E\)](#), [Wooldridge \(2010, chap. 12.7\)](#), and [Gould, Pitblado, and Poi \(2010, chap. 1\)](#). GMM is covered in [Cameron and Trivedi \(2005, chap. 5\)](#), [Greene \(2018, chap. 13\)](#), and [Wooldridge \(2010, chap. 14\)](#).

16.10 Exercises

1. Consider estimation of the logit model covered in section 10.5 and chapter 17. Then $Q(\beta) = \sum_i \{y_i \ln \Lambda_i + (1 - y_i)\Lambda_i\}$, where $\Lambda_i = \Lambda(\mathbf{x}'_i \beta) = \exp(\mathbf{x}'_i \beta) / \{1 + (\mathbf{x}'_i \beta)\}$. Show that $g(\beta) = \sum_i (y_i - \Lambda_i)\mathbf{x}_i$ and $H(\beta) = \sum_i -\Lambda_i(1 - \Lambda_i)\mathbf{x}_i\mathbf{x}'_i$. Hint: $\partial \Lambda(z) / \partial z = \Lambda(z)\{1 - \Lambda(z)\}$. Use the data on `docvis` to generate the binary variable `d_dv` for whether there are any doctor visits. Using just 2002 data, as in this chapter, use `logit` to perform logistic regression of the binary variable `d_dv` on `private`, `chronic`, `female`, `income`, and an intercept. Obtain robust estimates of the standard errors. You should find that the coefficient of `private`, for example, equals 1.27266, with a robust standard error of 0.0896928.
2. Adapt the code of section 16.2.3 to fit the logit model of exercise 1 using NR iterations coded in Mata. Hint: In defining an $n \times 1$ column vector with entries Λ_i , you may find it helpful to use the fact that `J(n, 1, 1)` creates an $n \times 1$ vector of 1s.
3. Adapt the code of section 16.5.3 to fit the logit model of exercise 1 using the `m1` command method `lf`.
4. Generate 100,000 observations from the following logit model DGP,

$$y_i = 1 \text{ if } \beta_1 + \beta_2 x_i + u_i > 0 \text{ and } y_i = 0 \text{ otherwise}; \quad (\beta_1, \beta_2) = (0, 1); \quad x_i \sim N(0, 1)$$

where u_i is logistically distributed. Using the inverse transformation method, you can compute a draw u from the logistic distribution as
 $u = -\ln\{(1 - r)/r\}$, where r is a draw from the uniform distribution. Use data from this DGP to check the consistency of your estimation method in exercise 3 or, more simply, of the `logit` command.

5. Consider the NLS example in section 16.5.5 with an exponential conditional mean. Fit the model using the `m1` command and the `lfnls` program. Also, fit the model using the `n1` command, given in section 13.3.6. Verify that these two methods give the same parameter estimates but, as noted in the text, the robust standard errors differ.
6. Continue the preceding exercise. Fit the model using the `m1` command and the `lfnls` program with default standard errors. These implicitly assume that the NLS model error has a variance of $\sigma^2 = 1$. Obtain an estimate of $s^2 = (1/N - K) \sum_i \{y_i - \exp(\mathbf{x}'_i \hat{\beta})\}^2$, using the `predictnl` postestimation command to obtain $\exp(\mathbf{x}'_i \hat{\beta})$. Then, obtain an estimate of the VCE by

multiplying the stored result `e(v)` by s^2 . Obtain the standard error of $\hat{\beta}_{\text{private}}$, and compare this with the standard error obtained when the NLS model is fit using the `n1` command with a default estimate of the VCE.

7. Consider a Poisson regression of `docvis` on the regressors `private`, `chronic`, `female`, and `income` and the programs given in section [16.7](#). Run the `ml model d0 d0pois` command, and confirm that you get the same output as produced by the code in section [16.7.3](#). Confirm that the nonrobust standard errors are the same as those obtained using `poisson` with default standard errors. Run `ml model d1 d1pois`, and confirm that you get the same output as produced by the code in section [16.7.3](#). Run `ml model d2 d2pois`, and confirm that you get the same output as that given in section [16.7.3](#).
8. Adapt the code of section [16.7.3](#) to fit the logit model of exercise 1 by using `ml` command method `d0`.
9. Adapt the code of section [16.7.2](#) to fit the logit model of exercise 1 by using `ml` command method `lf1` with robust standard errors reported.
10. Adapt the code of section [16.7.3](#) to fit the logit model of exercise 1 by using `ml` command method `d2`.
11. Consider the negative binomial example given in section [16.5.4](#). Fit this same model by using the `ml` command method `d0`. Hint: See the Weibull example in [R] `ml`.

Chapter 17

Binary outcome models

17.1 Introduction

Regression analysis of a qualitative binary or dichotomous variable is a commonplace problem in applied statistics. Models for mutually exclusive binary outcomes focus on the determinants of the probability p of the occurrence of one outcome rather than an alternative outcome that occurs with a probability of $1 - p$. An example where the binary variable is of direct interest is modeling whether an individual has insurance. In regression analysis, we want to measure how the probability p varies across individuals as a function of regressors. A different type of example is predicting the propensity score p , the conditional probability of participation (rather than nonparticipation) of an individual in a treatment program. In the treatment-effects literature, this prediction given observable variables is an important intermediate step, even though ultimate interest lies in outcomes of that treatment.

For regression with a continuous dependent variable, the standard model is the linear model. For binary outcome data, there are two standard parametric models—the logit model and the probit model. These specify different functional forms for p as a function of regressors, and the models are fit by maximum likelihood (ML). These models are nonlinear, making direct interpretation of parameters more difficult. The resulting marginal effects (MES) and predicted probabilities, however, are in practice similar across the two models. The logit model is the most commonly used model for binary outcomes in applied statistics, while economists additionally use the probit model. The linear probability model (LPM), fit by ordinary least squares (OLS), is also used at times.

The logit and probit models were introduced in chapter 10 as examples of nonlinear regression. This chapter repeats some material from chapter 10 while also providing further detail on the estimation and interpretation of cross-sectional binary outcome models. Because of their nonlinear functional forms, focus in interpretation is on MES of regressors rather than their coefficients. Extensions of the standard model include endogenous regressors and clustered and grouped data.

17.2 Some parametric models

Different binary outcome models have a common structure. The dependent variable, y_i , takes only two values, so its distribution is unambiguously the Bernoulli, or binomial with one trial, with a probability of p_i . Logit and probit models correspond to different regression models for p_i .

17.2.1 Basic model

Suppose the outcome variable, y , takes one of two values:

$$y = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1 - p \end{cases} \quad (17.1)$$

Given our interest in modeling p as a function of regressors \mathbf{x} , there is no loss of generality in setting the outcome values to 1 and 0. The probability mass function for the observed outcome, y , is $p^y(1-p)^{1-y}$, with $E(y) = p$ and $\text{Var}(y) = p(1-p)$.

A regression model is formed by parameterizing p to depend on an index function $\mathbf{x}'\boldsymbol{\beta}$, where \mathbf{x} is a $K \times 1$ regressor vector and $\boldsymbol{\beta}$ is a vector of unknown parameters. In standard binary outcome models, the conditional probability has the form

$$p_i \equiv \Pr(y_i = 1 | \mathbf{x}) = F(\mathbf{x}'_i \boldsymbol{\beta})$$

where $F(\cdot)$ is a specified parametric function of $\mathbf{x}'\boldsymbol{\beta}$, usually a cumulative distribution function (c.d.f.) on $(-\infty, \infty)$ because this ensures that the bounds $0 \leq p \leq 1$ are satisfied. Note that in this model,

$$\begin{aligned} E(y_i | \mathbf{x}_i) &= F(\mathbf{x}' \boldsymbol{\beta}) \\ \text{Var}(y_i | \mathbf{x}_i) &= F(\mathbf{x}' \boldsymbol{\beta}) \{1 - F(\mathbf{x}' \boldsymbol{\beta})\} \end{aligned} \quad (17.2)$$

17.2.2 Logit, probit, linear probability, and complementary log–log models

Models differ in the choice of function, $F(\cdot)$. Four commonly used functional forms for $F(\mathbf{x}'\boldsymbol{\beta})$, shown in table 17.1, are the logit, probit, linear probability, and complementary log–log forms.

Table 17.1. Four commonly used binary outcome models

Model	Probability $p = \Pr(y = 1 \mathbf{x})$	ME $\partial p / \partial x_j$
Logit	$\Lambda(\mathbf{x}'\boldsymbol{\beta}) = e^{\mathbf{x}'\boldsymbol{\beta}} / (1 + e^{\mathbf{x}'\boldsymbol{\beta}})$	$\Lambda(\mathbf{x}'\boldsymbol{\beta})\{1 - \Lambda(\mathbf{x}'\boldsymbol{\beta})\}\beta_j$
Probit	$\Phi(\mathbf{x}'\boldsymbol{\beta}) = \int_{-\infty}^{\mathbf{x}'\boldsymbol{\beta}} \phi(z)dz$	$\phi(\mathbf{x}'\boldsymbol{\beta})\beta_j$
Complementary log–log	$C(\mathbf{x}'\boldsymbol{\beta}) = 1 - \exp\{-\exp(\mathbf{x}'\boldsymbol{\beta})\}$	$\exp\{-\exp(\mathbf{x}'\boldsymbol{\beta})\} \exp(\mathbf{x}'\boldsymbol{\beta})\beta_j$
Linear probability	$F(\mathbf{x}'\boldsymbol{\beta}) = \mathbf{x}'\boldsymbol{\beta}$	β_j

The logit model specifies that $F(\cdot) = \Lambda(\cdot)$, the c.d.f. of the logistic distribution. The probit model specifies that $F(\cdot) = \Phi(\cdot)$, the standard normal c.d.f. Logit and probit functions are symmetric around zero and are widely used in microeconomics. The complementary log–log model is asymmetric around zero. Its use is sometimes recommended when the distribution of y is skewed such that there is a high proportion of either zeros or ones in the dataset. The LPM corresponds to linear regression and does not impose the restriction that $0 \leq p \leq 1$.

The last column in the table gives expressions for the corresponding MES, used in section 17.6, where $\phi(\cdot)$ denotes the standard normal density.

17.2.3 Latent-variable interpretation and identification

Binary outcome models can be given a latent-variable interpretation, though this is not necessary. The advantage of doing so is that it provides a link with the linear regression model, explains more deeply the difference between logit and probit models, and provides the basis for extension to some multinomial models given in chapter 18.

We distinguish between the observed binary outcome, y , and an underlying continuous unobservable (or latent) variable, y^* , that satisfies the single-index model

$$y^* = \mathbf{x}'\boldsymbol{\beta} + u \quad (17.3)$$

Although y^* is not observed, we do observe

$$y = \begin{cases} 1 & \text{if } y^* > 0 \\ 0 & \text{if } y^* \leq 0 \end{cases} \quad (17.4)$$

where the zero threshold is a normalization that is of no consequence if \mathbf{x} includes an intercept.

Given the latent-variable models (17.3) and (17.4), we have

$$\begin{aligned} \Pr(y = 1) &= \Pr(\mathbf{x}'\boldsymbol{\beta} + u > 0) \\ &= \Pr(u > -\mathbf{x}'\boldsymbol{\beta}) \\ &= 1 - \Pr(u \leq -\mathbf{x}'\boldsymbol{\beta}) \\ &= \Pr(u < \mathbf{x}'\boldsymbol{\beta}) \\ &= F(\mathbf{x}'\boldsymbol{\beta}) \end{aligned}$$

where the second to last equality assumes that u is symmetrically distributed about zero and $F(\cdot)$ is the c.d.f. of u . This yields the probit model if u is standard normally distributed and the logit model if u is logistically distributed.

Identification of the latent-variable model requires that we fix its scale by placing a restriction on the variance of u , because the single-index model can identify $\boldsymbol{\beta}$ only up to scale. An explanation for this is that we observe only whether $y^* = \mathbf{x}'\boldsymbol{\beta} + u > 0$. But this is not distinguishable from the outcome $\mathbf{x}'\boldsymbol{\beta}^+ + u^+ > 0$, where $\boldsymbol{\beta}^+ = a\boldsymbol{\beta}$ and $u^+ = au$ for any $a > 0$. We

can identify only β/σ , where σ is the standard deviation (scale parameter) of u .

To uniquely define the scale of β , we set $\sigma = 1$ in the probit model and $\pi/\sqrt{3}$ in the logit model. Thus, β is scaled differently in the two models; see section [17.4.4](#).

17.3 Estimation

For parametric models with exogenous covariates, the maximum likelihood estimator (MLE) is the natural estimator because the density is unambiguously the Bernoulli. Stata provides ML procedures for logit, probit, and complementary log–log models and for several variants of these models. For models with endogenous covariates, instrumental-variables (IV) methods can instead be used; see section [17.9](#).

17.3.1 ML estimation

For binary models other than the LPM, estimation is by ML. This ML estimation is straightforward. The density for a single observation can be compactly written as $p_i^{y_i} (1 - p_i)^{1-y_i}$, where $p_i = F(\mathbf{x}'_i \boldsymbol{\beta})$. For a sample of N independent observations, the MLE, $\hat{\boldsymbol{\beta}}$, maximizes the associated log-likelihood function

$$Q(\boldsymbol{\beta}) = \sum_{i=1}^N [y_i \ln F(\mathbf{x}'_i \boldsymbol{\beta}) + (1 - y_i) \ln\{1 - F(\mathbf{x}'_i \boldsymbol{\beta})\}] \quad (17.5)$$

The MLE is obtained by iterative methods and is asymptotically normally distributed.

Consistent estimates of $\boldsymbol{\beta}$ are obtained if $F(\mathbf{x}'_i \boldsymbol{\beta})$ is correctly specified. When instead it is misspecified, because the functional form $F(\cdot)$ is misspecified or because $\mathbf{x}'_i \boldsymbol{\beta}$ is misspecified, quasi-ML theory applies. Thus, we use robust standard errors that nonetheless provide consistent estimates of the variance of $\hat{\boldsymbol{\beta}}$.

17.3.2 The logit and probit commands

The syntax for the `logit` command is

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

The syntaxes for the `probit` and `cloglog` commands are similar.

Like the `regress` command, available options include `vce(cluster clustvar)` and `vce(robust)` for variance estimation. The constant is included by default but can be suppressed by using the `noconstant` option.

The `or` option of `logit` presents exponentiated coefficients. The rationale is that for the logit model, the log of the odds ratio $\ln\{p/(1 - p)\}$ can be shown to be linear in \mathbf{x} and $\boldsymbol{\beta}$. It follows that the odds ratio $p/(1 - p) = \exp(\mathbf{x}'\boldsymbol{\beta})$, so that e^{β_j} measures the multiplicative effect of a unit change in regressor x_j on the odds ratio. Thus, many researchers prefer logit coefficients to be reported after exponentiation, that is, as e^β rather than β . Alternatively, the `logistic` command estimates the parameters of the logit model and directly reports the exponentiated coefficients.

17.3.3 Robust estimate of the variance–covariance matrix of the estimator

Binary outcome models are unusual in that in theory there is no advantage in using the robust sandwich form for the variance–covariance matrix of the estimator (VCE) of the MLE if data are independent over i and $F(\mathbf{x}'\boldsymbol{\beta})$ is correctly specified. The reason is that the ML default standard errors are obtained by imposing the restriction $\text{Var}(y|\mathbf{x}) = F(\mathbf{x}'\boldsymbol{\beta})\{1 - F(\mathbf{x}'\boldsymbol{\beta})\}$, and this must necessarily hold because the variance of a binary variable is always $p(1 - p)$; see [Cameron and Trivedi \(2005, 469\)](#) for further explanation. If $F(\mathbf{x}'\boldsymbol{\beta})$ is correctly specified, the `vce(robust)` option is not required.

In practice, $F(\mathbf{x}'\boldsymbol{\beta})$ is likely to be misspecified because the functional form $F(\cdot)$ is misspecified or because $\mathbf{x}'_i\boldsymbol{\beta}$ is misspecified. Then quasi-ML theory applies (see section 13.3.1), and we need to use robust standard errors that nonetheless provide consistent estimates of the variance of $\hat{\boldsymbol{\beta}}$. So we generally obtain robust standard errors.

Note also that for binary outcomes in the special case of independent observations we may infer that the functional form $F(\mathbf{x}'\boldsymbol{\beta})$ is misspecified if the `vce(robust)` option produces substantially different variances from the default.

When observations are dependent because of clustering, the appropriate option is to use `vce(cluster clustvar)`, even if $F(\mathbf{x}'\boldsymbol{\beta})$ is correctly specified.

17.3.4 OLS estimation of linear probability model

If $F(\cdot)$ is assumed to be linear, that is, $p = \mathbf{x}'\boldsymbol{\beta}$, then the linear conditional mean function defines the LPM. The LPM can be consistently fit by OLS regression of y on \mathbf{x} using `regress`. A major limitation of the method, however, is that the fitted values $\mathbf{x}'\hat{\boldsymbol{\beta}}$ will not necessarily be in the $[0, 1]$ interval. And because $\text{Var}(y|\mathbf{x}) = (\mathbf{x}'\boldsymbol{\beta})(1 - \mathbf{x}'\boldsymbol{\beta})$ for the LPM, the regression is inherently heteroskedastic, so a robust estimate of the VCE should be used.

17.3.5 Estimation with proportions or fractional response data

Proportions data or fractional response data are continuous data that lie between zero and one, such as county-level data on the proportion of people with private insurance or on the fraction of income spent on food.

Because y lies between zero and one, its conditional mean should lie between zero and one. A natural model is then $E(y_i|\mathbf{x}_i) = F(\mathbf{x}'_i\boldsymbol{\beta})$, where $F(\cdot)$ is the function corresponding to a binary outcome model such as the logit or the probit model. This model can be fit by nonlinear least squares (NLS); an `n1` command for probit is given in section 10.6.2. Inference should be based on robust standard errors because the error in the NLS model is unlikely to be homoskedastic.

Efficiency gains may be possible by additionally modeling the heteroskedasticity. A natural starting point is $\text{Var}(y_i|\mathbf{x}_i) = F(\mathbf{x}'_i\boldsymbol{\beta}) \times \{1 - F(\mathbf{x}'_i\boldsymbol{\beta})\}$, as in (17.2). One might use feasible generalized NLS. Simpler is to use the `fracreg` command, which

uses the same objective function ([17.5](#)) as that for a binary outcome MLE. In either case, robust standard errors need to be used to guard against misspecification of the model for heteroskedasticity.

Further discussion and examples are given in section [17.10](#).

17.4 Example

We analyze data on supplementary health insurance coverage. Initial analysis estimates the parameters of the models of section [17.2](#).

17.4.1 Data description

The data come from wave 5 (2002) of the Health and Retirement Study (HRS), a panel survey sponsored by the National Institute of Aging. The sample is restricted to Medicare beneficiaries. The HRS contains information on a variety of medical service uses. The elderly can obtain supplementary insurance coverage either by purchasing it themselves or by joining employer-sponsored plans. We use the data to analyze the purchase of private insurance (`ins`) from any source, including private markets or associations. The insurance coverage broadly measures both individually purchased and employer-sponsored private supplementary insurance and includes Medigap plans and other policies.

The explanatory variables include health status, socioeconomic characteristics, and spouse-related information. Self-assessed health-status information is used to generate a dummy variable (`hstatusg`) that measures whether health status is good, very good, or excellent. Other measures of health status are the number of limitations (up to five) on activities of daily living (`adl`) and the total number of chronic conditions (`chronic`). Socioeconomic variables used are age, gender, race, ethnicity, marital status, years of education, and retirement status (respectively, `age`, `female`, `white`, `hisp`, `married`, `educyear`, `retire`); household income (`hhincome`); and log household income if positive (`linc`). Spouse retirement status (`sretire`) is an indicator variable equal to 1 if a retired spouse is present.

For conciseness, we use global macros to create variable lists, presenting the variables used in sections [17.4–17.6](#) followed by additional variables used in section [17.9](#). We have

```

. * Read in data, define globals, and summarize key variables
. qui use mus217hrs
. global xlist age hstatusg hhincome educyear married hisp
. global extralist female white chronic adl sretire
. summarize ins retire $xlist $extralist

```

Variable	Obs	Mean	Std. dev.	Min	Max
ins	3,206	.3870867	.4871597	0	1
retire	3,206	.6247661	.4842588	0	1
age	3,206	66.91391	3.675794	52	86
hstatusg	3,206	.7046163	.4562862	0	1
hhincome	3,206	45.26391	64.33936	0	1312.124
educyear	3,206	11.89863	3.304611	0	17
married	3,206	.7330006	.442461	0	1
hisp	3,206	.0726762	.2596448	0	1
female	3,206	.477854	.4995872	0	1
white	3,206	.8206488	.383706	0	1
chronic	3,206	2.063319	1.416434	0	8
adl	3,206	.301622	.8253646	0	5
sretire	3,206	.3883344	.4874473	0	1

17.4.2 Logit regression

We begin with ML estimation of the logit model, with heteroskedastic–robust standard errors computed.

```

. * Logit regression
. logit ins retire $xlist, vce(robust)

Iteration 0:  log pseudolikelihood = -2139.7712
Iteration 1:  log pseudolikelihood = -1996.7434
Iteration 2:  log pseudolikelihood = -1994.8864
Iteration 3:  log pseudolikelihood = -1994.8784
Iteration 4:  log pseudolikelihood = -1994.8784

Logistic regression                                         Number of obs = 3,206
                                                               Wald chi2(7) = 256.49
                                                               Prob > chi2 = 0.0000
                                                               Pseudo R2 = 0.0677

Log pseudolikelihood = -1994.8784

```

ins	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
retire	.1969297	.0849524	2.32	0.020	.0304261	.3634332
age	-.0145955	.0110189	-1.32	0.185	-.0361921	.007001
hstatusg	.3122654	.0918378	3.40	0.001	.1322667	.4922641
hhincome	.0023036	.0011485	2.01	0.045	.0000526	.0045546
educyear	.1142626	.0143616	7.96	0.000	.0861143	.1424108
married	.578636	.0941579	6.15	0.000	.39409	.7631821
hisp	-.8103059	.1936793	-4.18	0.000	-1.18991	-.4307016
_cons	-1.715578	.7279873	-2.36	0.018	-3.142407	-.2887495

All regressors other than `age` are statistically significantly different from zero at the 0.05 level. For the logit model, the sign of the coefficient is also the sign of the ME. Further discussion of these results is deferred to the next section, where we compare logit parameter estimates with those from other models.

The iteration log shows fast convergence in four iterations. Later output suppresses the iteration log to save space. In actual empirical work, it is best to keep the log. For example, many iterations may signal a high degree of multicollinearity.

17.4.3 Coefficient interpretation

The logit (and probit) models are single-index models (see section 13.7.3), so if one coefficient is twice as big as the other, then the effect of a unit change in the corresponding regressor is twice as big as the other. Furthermore, the sign of the coefficient gives the sign of the effect because $F'(\cdot) > 0$.

Thus, one more year of education is associated with an increase in the probability of having private insurance. And being married also has a positive effect that is roughly equivalent to the effect of 5 more years of schooling because $0.5786/0.1143 = 5.06$.

MES are presented in section [17.6](#). A rough rule of thumb for the logit model is that the average marginal effect (AME) of a change in the j th regressor equals $\bar{y}(1 - \bar{y})\beta_j$. So one more year of education is associated with an increase of $0.387 \times 0.613 \times 0.114 = 0.027$ of having higher insurance. For the probit model, a similar approximation is that the ME is at most $0.4 \times \beta_j$.

Finally, from section [17.3.2](#), the logit estimates imply that one more year of education is associated with an $e^{0.114}$ times higher odds ratio (of having private insurance versus not having private insurance).

17.4.4 Comparison of binary models and parameter estimates

It is well known that logit and probit models have similar shapes for central values of $F(\cdot)$ but differ in the tails as $F(\cdot)$ approaches 0 or 1. At the same time, the corresponding coefficient estimates from the two models are scaled quite differently. It is an elementary mistake to suppose that the different models have different implications simply because the estimated coefficients across models are different. However, this difference is mainly due to different functional forms for the probabilities. The MES and predicted probabilities, presented in sections [17.5](#) and [17.6](#), are much more similar across models.

Coefficients can be compared across models, using the following rough conversion factors ([Amemiya 1981](#), 1,488):

$$\begin{aligned}\widehat{\boldsymbol{\beta}}_{\text{Logit}} &\simeq 4\widehat{\boldsymbol{\beta}}_{\text{OLS}} \\ \widehat{\boldsymbol{\beta}}_{\text{Probit}} &\simeq 2.5\widehat{\boldsymbol{\beta}}_{\text{OLS}} \\ \widehat{\boldsymbol{\beta}}_{\text{Logit}} &\simeq 1.6\widehat{\boldsymbol{\beta}}_{\text{Probitt}}\end{aligned}$$

The motivation is that it is better to compare the ME, $\partial p/\partial x_j$, across models, and it can be shown that $\partial p/\partial x_j \leq 0.25\widehat{\beta}_j$ for logit, $\partial p/\partial x_j \leq 0.4\widehat{\beta}_j$ for probit, and $\partial p/\partial x_j = \widehat{\beta}_j$ for OLS. The greatest departures across the models occur in the tails.

We estimate the parameters of the logit and probit models by ML and the LPM by OLS, computing standard errors and z statistics based on both default and robust estimates of the VCE. The following code saves results for each model with the `estimates store` command.

```
. * Comparison of estimates for logit, probit and LPM models
. qui logit ins retire $xlist
. estimates store blogit
. qui probit ins retire $xlist
. estimates store bprobit
. qui regress ins retire $xlist
. estimates store bols
. qui logit ins retire $xlist, vce(robust)
. estimates store blogitr
. qui probit ins retire $xlist, vce(robust)
. estimates store bprobitr
. qui regress ins retire $xlist, vce(robust)
. estimates store bolsr
```

This leads to the following output table of parameter estimates across the models:

```
. * Table for comparing models
. estimates table blogit blogitr bprobit bprobitr bols bolsr,
>      t stats(N ll) b(%7.3f) stfmt(%8.2f) eq(1)
```

Variable	blogit	blogitr	bprobit	bprobitr	bols	bolsr
retire	0.197 2.34	0.197 2.32	0.118 2.31	0.118 2.30	0.041 2.24	0.041 2.24
age	-0.015 -1.29	-0.015 -1.32	-0.009 -1.29	-0.009 -1.32	-0.003 -1.20	-0.003 -1.25
hstatusg	0.312 3.41	0.312 3.40	0.198 3.56	0.198 3.57	0.066 3.37	0.066 3.45
hhincome	0.002 3.02	0.002 2.01	0.001 3.19	0.001 2.21	0.000 3.58	0.000 2.63
educyear	0.114 8.05	0.114 7.96	0.071 8.34	0.071 8.33	0.023 8.15	0.023 8.63
married	0.579 6.20	0.579 6.15	0.362 6.47	0.362 6.46	0.123 6.38	0.123 6.62
hisp	-0.810 -4.14	-0.810 -4.18	-0.473 -4.28	-0.473 -4.36	-0.121 -3.59	-0.121 -4.49
_cons	-1.716 -2.29	-1.716 -2.36	-1.069 -2.33	-1.069 -2.40	0.127 0.79	0.127 0.83
N	3206 -1994.88	3206 -1994.88	3206 -1993.62	3206 -1993.62	3206 -2104.75	3206 -2104.75

Legend: b/t

The coefficients across the models tell a qualitatively similar story about the impact of a regressor on $\text{Pr}(\text{ins} = 1)$. The rough rules for parameter conversion also stand up reasonably well because the logit estimates are roughly five times the OLS estimates and the probit estimates are roughly three times the OLS coefficients. The standard errors are similarly rescaled, so that the reported z statistics for the coefficients are similar across the three models. For the logit and probit coefficients, the robust and default z statistics are quite similar, aside from those for the `hhincome` variable. For OLS, which does not allow for the intrinsic heteroskedasticity of binary data, there is a bigger difference.

In section [17.5](#), we will see that the fitted probabilities are similar for the logit and probit specifications. The linear functional form does not constrain the fitted values to the $[0, 1]$ interval, however, and we find differences in the fitted-tail values between the LPM and the logit and probit models.

17.4.5 Wald tests

Tests on coefficients of variables are most easily performed by using the `test` command, which implements a Wald test. For example, we may test for the presence of interaction effects with `age`, with all regressors aside from `retire` interacted with `age`. The null hypothesis is that the coefficients of the additional regressors are all zero because then there are no interaction effects.

To avoid the need to create new variables, we use the factor-variable binary operator `#` to define the interaction variables. To perform the subsequent Wald test, we use the `testparm` command rather than `test`. We obtain

```
. * Wald test for no interactions
. global intlist c.age#c.age c.age#i.hstatusg c.age#c.hhincome
>      c.age#c.educyear c.age#i.married c.age#i.hisp
. qui logit ins retire $xlist $intlist, vce(robust) nolog
. testparm $intlist
( 1) [ins]c.age#c.age = 0
( 2) [ins]1.hstatusg#c.age = 0
( 3) [ins]c.age#c.hhincome = 0
( 4) [ins]c.age#c.educyear = 0
( 5) [ins]1.married#c.age = 0
( 6) [ins]1.hisp#c.age = 0
chi2(  6) =    17.16
Prob > chi2 =    0.0087
```

The *p*-value is 0.0087, so the null hypothesis is rejected at the 0.01 level.

17.4.6 Likelihood-ratio tests

A likelihood-ratio (LR) test (see section 11.4) provides an alternative method for testing hypotheses, provided default standard errors are used. It is asymptotically equivalent to the Wald test based on default standard errors if the model is correctly specified.

To implement the LR test of the preceding hypothesis, we estimate parameters of both the general and the restricted models with default estimates of the VCE and then use the `lrtest` command. We obtain

```

. * LR test for no interactions
. qui logit ins retire $xlist $intlist
. estimates store B
. qui logit ins retire $xlist
. lrtest B

Likelihood-ratio test
Assumption: . nested within B
LR chi2(6) = 16.10
Prob > chi2 = 0.0132

```

This test has a p -value of 0.0132, so the null hypothesis is rejected at the 0.05 level but not at the 0.01 level. Wald and LR tests do not yield identical results.

In some situations, the main focus is on the predicted probability of the model, and the sign and size of the coefficients are not the focus of the inquiry. An example is the estimation of propensity scores, in which case a recommendation is often made to saturate the model and then to choose the best model by using the Bayesian information criterion (BIC).

17.4.7 Model comparison

A question often arises: which model is better, logit or probit? As will be seen in the next section, in many cases, the fitted probability is very similar over a large part of the range of $\mathbf{x}'\boldsymbol{\beta}$. Larger differences may be evident in the tails of the distribution, but a large sample is required to reliably differentiate between models on the basis of tail behavior.

Because logit and probit models are nonnested, a penalized likelihood criterion such as Akaike's information criterion or BIC (see section 13.8.2) is appealing for model selection. However, these two models have the same number of parameters, so this reduces to choosing the model with the higher log likelihood. The probit model has a log likelihood of $-1,993.62$ (see the table in section [17.4.4](#)), which is 1.26 higher than the $-1,994.88$ for logit. This favors the probit model, but the difference is not great. For example, an LR test of a single restriction rejects at the 0.05 level if the LR statistic exceeds 3.84 or, equivalently, if the change in log likelihood is $3.84/2 = 1.92$.

17.4.8 Generalized linear model estimation

The logit and probit models are examples of generalized linear models, and the `glm` command can yield identical results to the `logit` and `probit` commands.

For example, to obtain logit ML estimates with heteroskedastic-robust standard errors, we give the command

```
. * Logit command results duplicated using the glm command
. glm ins $xlist, link(logit) family(binomial) vce(robust)
(output omitted)
```

If the `family(binomial)` option is dropped, we instead obtain NLS estimates for the logit model. We have

```
. * NLS estimation of logit model using the glm command
. glm ins retire $xlist, link(logit) vce(robust) nolog

Generalized linear models
Optimization : ML
Number of obs = 3,206
Residual df = 3,198
Scale parameter = .2181226
Deviance = 697.5562266
(1/df) Deviance = .2181226
Pearson = 697.5562266
(1/df) Pearson = .2181226
Variance function: V(u) = 1 [Gaussian]
Link function : g(u) = ln(u/(1-u)) [Logit]
AIC = 1.317671
BIC = -25119.19
Log pseudolikelihood = -2104.227411
```

ins	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
retire	.2079366	.086513	2.40	0.016	.0383743	.3774988
age	-.01465	.0107263	-1.37	0.172	-.0356731	.0063732
hstatusg	.2473319	.0907609	2.73	0.006	.0694438	.42522
hhincome	.0045752	.0020291	2.25	0.024	.0005983	.0085522
educyear	.0994968	.0151952	6.55	0.000	.0697147	.1292788
married	.4839629	.1001032	4.83	0.000	.2877643	.6801616
hisp	-.7191678	.1941781	-3.70	0.000	-1.09975	-.3385857
_cons	-1.495278	.7061223	-2.12	0.034	-2.879252	-.111304

The estimated coefficients are generally within 20% of the logit estimates and have slightly higher standard errors.

This last estimator can be used with proportions data and yields fitted values of the dependent variable that necessarily lie between zero and one; see sections [17.3.5](#) and [17.10](#).

17.5 Goodness of fit and prediction

The Stata output for the logit and probit regressions has a similar format. The log likelihood and the LR test of the joint significance of the regressors and its p -value are given. However, some measures of overall goodness of fit are desirable, including those that are specific to the binary outcome model.

Three approaches to evaluating the fit of the model are pseudo- R^2 measures, comparisons of group-average predicted probabilities with sample frequencies, and comparisons based on classification (\hat{y} equals zero or one). None of these is the most preferred measure a priori. Below, we discuss comparisons of model fit using predicted probabilities.

17.5.1 Pseudo- R^2 measures

In linear regression, the total sum of squared deviations from the mean can be decomposed into explained and residual sums of squares, and R^2 measures the ratio of explained sum of squares to total sum of squares, with 0 and 1 as the lower and upper limits, respectively. These properties do not carry over to nonlinear regression. Yet there are some measures of fit that attempt to mimic the R^2 measure of linear regression. There are several R^2 measures, one of which is included in the Stata output.

McFadden's \tilde{R}^2 is computed as $1 - L_N(\hat{\beta})/L_N(\bar{y})$, where $L_N(\hat{\beta})$ denotes the maximized or fitted log-likelihood value and $L_N(\bar{y})$ denotes the value of the log likelihood in the intercept-only model. When applied to models with binary and multinomial outcomes, the lower and upper bounds of the pseudo- R^2 measure are 0 and 1 (see section 13.8.1), though McFadden's \tilde{R}^2 is not a measure of the proportion of variance of the dependent variable explained by the model. For the fitted logit model, $\tilde{R}^2 = 0.068$.

17.5.2 Comparing predicted frequencies with sample frequencies

In-sample comparison of the average predicted probabilities, $N^{-1} \sum \hat{p}_i$, with the sample frequency, \bar{y} , is not helpful for evaluating the fit of binary

outcome models. In particular, the two are necessarily equal for logit models that include an intercept because the logit MLE first-order conditions can be shown to then impose this condition.

However, this comparison may be a useful thing to do for subgroups of observations. The Hosmer–Lemeshow specification test evaluates the goodness of fit by comparing the sample frequency of the dependent variable with the predicted frequency based on the fitted probability within subgroups of observations, with the number of subgroups being specified by the investigator. The null hypothesis is that the two are equal. The test is similar to the Pearson chi-squared goodness-of-fit test.

Let $\overline{\hat{p}_g}$ and \overline{y}_g denote, respectively, the average predicted probability and sample average frequency in group g . The test statistic is $\sum_{g=1}^G (\overline{\hat{p}_g} - \overline{y}_g)^2 / \overline{y}_g(1 - \overline{y}_g)$, where g is the group subscript. The groups are based on quantiles of the ordered predicted probabilities. For example, if $G = 10$, then each group corresponds to a decile of the ordered \hat{p}_i . Hosmer and Lemeshow established the null distribution by simulation. Under the null of correct specification, the statistic is distributed as $\chi^2(G - 2)$. However, two caveats should be noted: First, the test outcome is sensitive to the number of groups used in the specification. Second, much of what is known about the properties of the test is based on Monte Carlo evidence of the test's performance; see [Hosmer and Lemeshow \(1980\)](#) and [Hosmer, Lemeshow, and Sturdivant \(2013\)](#). Simulation evidence suggests that a fixed sample size specifying many groups in the test causes a divergence between the empirical c.d.f. and the c.d.f. of the $\chi^2(G - 2)$ distribution. For the correct form of the chi-squared goodness-of-fit test, see section 11.9.3.

The goodness-of-fit test is performed by using the `estat gof` postestimation command, which has the syntax

```
estat gof [if] [in] [weight] [, options]
```

where the `group(#)` option specifies the number of quantiles to be used to group the data, with 10 being the default. This test is highly parametric and can be used only if the initial model estimation uses default standard errors.

After estimating the parameters of the logit model, we perform this test, setting the number of groups to four. We obtain

```
. * Hosmer--Lemeshow goodness-of-fit test with 4 groups
. qui logit ins retire $xlist
. estat gof, group(4) table // Hosmer--Lemeshow goodness-of-fit test
note: obs collapsed on 4 quantiles of estimated probabilities.

Goodness-of-fit test after logistic model
Variable: ins
```

Table collapsed on quantiles of estimated probabilities

Group	Prob	Obs_1	Exp_1	Obs_0	Exp_0	Total
1	0.2849	146	161.2	656	640.8	802
2	0.3994	291	274.6	510	526.4	801
3	0.4779	387	355.0	415	447.0	802
4	0.9650	417	450.1	384	350.9	801

```
Number of observations = 3,206
Number of groups = 4
Hosmer-Lemeshow chi2(2) = 14.04
Prob > chi2 = 0.0009
```

The first row of the table, for example, shows that for the 802 observations with predicted probabilities less than the lower quartile predicted probability of 0.2849, there were 146 cases of $y = 1$, while 802 times the average predicted probability equals 161.2, an overprediction of 15.2. Across the four rows, the model overpredicts cases of $y = 1$ in the lower and upper quartiles of the predicted probabilities and underpredicts in the interquartile range. The outcome indicates misspecification because the p -value is 0.001.

To check whether the same outcome occurs if we use many groups to perform the test, we repeat the test for 10 groups.

```
. * Hosmer--Lemeshow goodness-of-fit test with 10 groups
. estat gof, group(10) // Hosmer--Lemeshow goodness-of-fit test
note: obs collapsed on 10 quantiles of estimated probabilities.

Goodness-of-fit test after logistic model
Variable: ins

Number of observations = 3,206
Number of groups = 10
Hosmer-Lemeshow chi2(8) = 31.48
Prob > chi2 = 0.0001
```

Again, the test rejects the maintained specification, this time with an even smaller p -value.

17.5.3 Comparing predicted outcomes with actual outcomes

The preceding measure is based on the fitted probability of having private insurance. We may instead want to predict the outcome itself, that is, whether an individual has private insurance ($\hat{y} = 1$) or does not have insurance ($\hat{y} = 0$). Strictly speaking, this depends upon a loss function. If we assume a symmetric loss function, then it is natural to set $\hat{y} = 1$ if $F(\mathbf{x}'\boldsymbol{\beta}) > 0.5$ and $\hat{y} = 0$ if $F(\mathbf{x}'\boldsymbol{\beta}) \leq 0.5$. One measure of goodness of fit is the percentage of correctly classified observations.

Goodness-of-fit measures based on classification can be obtained by using the `estat classification` postestimation command.

For the fitted logit model, we obtain

```

. * Comparing fitted probability and dichotomous outcome
. qui logit ins retire $xlist
. estat classification

```

Logistic model for ins

Classified	True		Total
	D	$\sim D$	
+	345	308	653
-	896	1657	2553
Total	1241	1965	3206

Classified + if predicted $\text{Pr}(D) \geq .5$
 True D defined as ins != 0

Sensitivity	$\text{Pr}(+ D)$	27.80%
Specificity	$\text{Pr}(- \sim D)$	84.33%
Positive predictive value	$\text{Pr}(D +)$	52.83%
Negative predictive value	$\text{Pr}(\sim D -)$	64.90%
False + rate for true $\sim D$	$\text{Pr}(+ \sim D)$	15.67%
False - rate for true D	$\text{Pr}(- D)$	72.20%
False + rate for classified +	$\text{Pr}(\sim D +)$	47.17%
False - rate for classified -	$\text{Pr}(D -)$	35.10%
Correctly classified		62.45%

The table presents a “confusion matrix” that compares fitted and actual values. The percentage of correctly specified values in this case is 62.45. In this example, 308 observations are misclassified as 1 when the correct classification is 0, and 896 values are misclassified as 0 when the correct value is 1. The remaining $345 + 1657$ observations are correctly specified.

The `estat classification` command also produces detailed output on classification errors, using terminology that is commonly used in biostatistics and is detailed in [R] **logistic postestimation**. The ratio $345/1241$, called the sensitivity measure, gives the fraction of observations with $y = 1$ that are correctly specified. The ratio $1657/1965$, called the specificity measure, gives the fraction of observations with $y = 0$ that are correctly specified. The ratios $308/1965$ and $896/1241$ are referred to, respectively, as the false positive and false negative classification error rates.

The classification statistics are less useful when models provide poor fit and the event of interest occurs with probability close to 0 or 1. For example, a logit model for whether an individual is unemployed ($y = 1$), a low-probability event, may yield predicted probabilities as less than 0.5 for all individuals, in which case all observations are classified as zero.

Machine-learning methods developed specifically for classification generally predict better than logit and probit models; section [28.6.7](#) provides a brief overview.

17.5.4 The predict command for fitted probabilities

Fitted probabilities can be computed by using the `pr` option of the `predict` postestimation command defined in section 13.5.1. The difference between logit and probit models may be small, especially over the middle portion of the distribution. On the other hand, the fitted probabilities from the LPM fit by OLS may be substantially different.

We first summarize the fitted probability from the three models that include only the `hhincome` variable as a regressor.

```
. * Calculate and summarize fitted probabilities for models with a single
> regressor
. qui logit ins hhincome
. predict plogit, pr
. qui probit ins hhincome
. predict pprobit, pr
. qui regress ins hhincome
. predict pols, xb
. summarize ins plogit pprobit pols
```

Variable	Obs	Mean	Std. dev.	Min	Max
ins	3,206	.3870867	.4871597	0	1
plogit	3,206	.3870867	.0787632	.3176578	.999738
pprobit	3,206	.3855051	.061285	.3349603	.9997945
pols	3,206	.3870867	.0724975	.3360834	1.814582

The mean and standard deviation are essentially the same in the three cases, but the range of the fitted values from the LPM includes six inadmissible

values outside the $[0, 1]$ interval. This fact should be borne in mind in evaluating the graph given below, which compares the fitted probability from the three models. The deviant observations from OLS stand out at the extremes of the range of the distribution, but the results for logit and probit cohere well.

For regressions with a single regressor, plotting predicted probabilities against that variable can be informative, especially if that variable takes a range of values. Such a graph illustrates the differences in the fitted values generated by different estimators. The example given below plots the fitted values from logit, probit, and LPM against household income (`hhincome`). For graph readability, the `jitter()` option is used to jitter the observed 0 and 1 values, leading to a band of outcome values that are around 0 and 1 rather than exactly 0 or 1.

```
. * Plot of predicted probabilities for models with a single regressor
. sort hhincome
. graph twoway (scatter ins hhincome, msize(vsmallest) jitter(3))
>     (line plogit hhincome, clstyle(p1))
>     (line pprobit hhincome, clstyle(p2))
>     (line pols hhincome, clstyle(p3)),
>     plotregion(style(none))
>     title("Predicted probabilities across models")
>     xtitle("Household income (hhincome)", size(medlarge)) xscale(titlegap(*5))
>     ytitle("Predicted probability", size(medlarge)) yscale(titlegap(*5))
>     legend(pos(1) ring(0) col(1)) legend(size(small))
>     legend(label(1 "Actual data (jittered)") label(2 "Logit"))
>     label(3 "Probit") label(4 "OLS"))
```

The first panel of figure 17.1 presents the results. The divergence between the first two and the LPM (OLS) estimates at high values of income stands out, though this is not necessarily serious because the number of observations in the upper range of income is quite small. The fitted values are close for most of the sample.

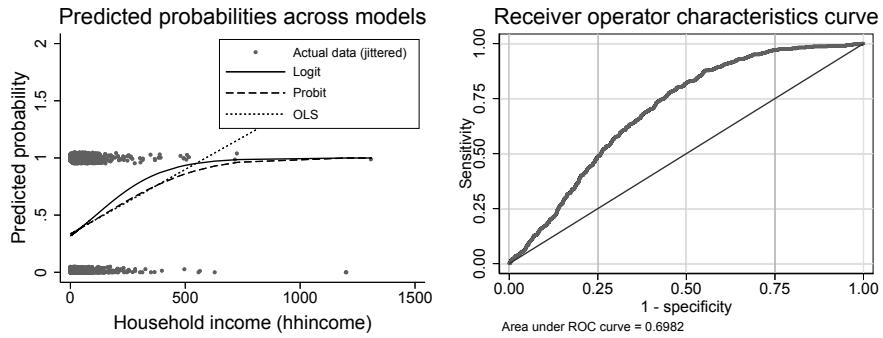


Figure 17.1. Predicted probabilities versus `hhincome` and receiver operator characteristics curve following logit

17.5.5 Receiver operator characteristics curve

Suppose a binary outcome model yields predicted probabilities \hat{p}_i and we classify $y_i = 1$ if $\hat{p}_i > c$, where the threshold c varies from 0 to 1. If $c = 1$, we predict $y_i = 0$ for all observations, so we never make the mistake of classifying $y_i = 1$ when in fact $y_i = 0$. And if $c = 0$, we predict $y_i = 1$ for all observations, so we will never make the mistake of classifying $y_i = 0$ when in fact $y_i = 1$.

The receiver operator characteristics (ROC) provides a graphical way to plot the tradeoff between the two types of error as c varies. Sensitivity measures the fraction of observations with $y = 1$ that are correctly specified, while specificity measures the fraction of observations with $y = 0$ that are correctly specified. The ROC curve plots sensitivity against $(1 - \text{specificity})$.

The `lroc` command produces the ROC curve following the `logit`, `probit`, and `ivprobit` commands. We do so for logit regression of `ins` on `income`, though `lroc` can follow regression with multiple regressors.

```
. * ROC curve following logit estimation
. qui logit ins hhincome
. lroc, lwidth(thick) title("Receiver operator characteristics curve")
>     msize(tiny)

Logistic model for ins

Number of observations =      3206
Area under ROC curve    =   0.6982
```

The second panel of figure 17.1 plots the ROC curve. The area under the curve (AUC) is literally the area under the ROC curve; higher values are better. A perfectly fitting model has sensitivity of 1 and specificity of 1, and the AUC is then 1. A model with no explanatory power has a ROC curve with the diagonal line and area equal to 0.5. The current model has AUC equal to 0.698.

17.5.6 The margins command for fitted probabilities

The `predict` command provides fitted probabilities for each individual, evaluating at $\mathbf{x} = \mathbf{x}_i$. At times, it is useful to instead obtain predicted probabilities at a representative value, $\mathbf{x} = \mathbf{x}^*$.

This can be done by using the `margins` command, presented in section 13.6. For a 65-year-old, married, retired non-Hispanic with good health status, 17 years of education, and an income equal to \$50,000 (so the `income` variable equals 50), we obtain

```
. * Fitted probabilities for selected baseline using margins
. qui logit ins retire $xlist, vce(robust)
. margins, at(age=65 retire=0 hstatusg=1 hhincome=50 educyear=17 married=1 hisp=0)
>     noatlegend
Adjusted predictions                                         Number of obs = 3,206
Model VCE: Robust
Expression: Pr(ins), predict()
```

	Delta-method				
	Margin	std. err.	z	P> z	[95% conf. interval]
_cons	.5705896	.0250894	22.74	0.000	.5214154 .6197638

The probability of having private insurance is 0.57 with the 95% confidence interval [0.52, 0.62].

Note that this reasonably tight confidence interval is for the probability that $y = 1$ given $\mathbf{x} = \mathbf{x}^*$. There is much more uncertainty in the outcome that $y = 1$ given $\mathbf{x} = \mathbf{x}^*$. For example, this difficulty in predicting actual values leads to the low \tilde{R}^2 for the logit model. This distinction is similar to that between predicting $E(y|\mathbf{x})$ and predicting $y|\mathbf{x}$ discussed in sections 4.2.5 and 13.5.2.

17.5.7 The prvalue command for fitted probabilities

An alternative is to use the community-contributed `prvalue` postestimation command ([Long and Freese 2014](#)), which has syntax

```
prvalue [ if ] [ in ] [ , x(conditions) rest(stat) options ]
```

where we list two key options. The `x(conditions)` option specifies the conditioning values of the regressors, and the default `rest(mean)` option specifies that the unconditioned variables be set at their sample averages. Omitting `x(conditions)` means that the predictions are evaluated at $\mathbf{x} = \bar{\mathbf{x}}$.

The `prvalue` command evaluated at the same regressor values yields

```
. * Fitted probabilities for selected baseline using prvalue
. qui logit ins retire $xlist, vce(robust)
. prvalue, x(age=65 retire=0 hstatusg=1 hhincome=50 educyear=17 married=1 hisp=0)
logit: Predictions for ins
Confidence intervals by delta method
Pr(y=1|x):          0.5706  [ 0.5214,      0.6198]
Pr(y=0|x):          0.4294  [ 0.3802,      0.4786]
         retire      age  hstatusg  hhincome  educyear  married      hisp
x=          0       65        1       50        17        1        0
```

The predicted probability and associated confidence interval are the same as those using the `margins` command.

17.6 Marginal effects

Three variants of MES, presented in section 13.7, are the AME, marginal effects at a representative value (MER), and marginal effects at the mean (MEM). In a nonlinear model, MES are more informative than coefficients.

The analytical formulas for the MES for the standard binary outcome models were given in table 17.1. For example, for the logit model, the ME with respect to a change in a continuous regressor, x_j , evaluated at $\mathbf{x} = \bar{\mathbf{x}}$, is estimated by $\Lambda(\bar{\mathbf{x}}' \hat{\boldsymbol{\beta}}) \{1 - \Lambda(\bar{\mathbf{x}}' \hat{\boldsymbol{\beta}})\} \hat{\beta}_j$. An associated confidence interval can be calculated by using the delta method.

17.6.1 AME

The AME is the average of MEs for each individual and is obtained as the default option of the `margins, dydx()` postestimation command. The associated standard errors and confidence interval for the AME are obtained using the delta method.

For the four binary regressors, we use the `i.` operator in the `logit` estimation command so that MES are computed using the finite-difference method, rather than the calculus method; see, for example, section 13.7.5. For the fitted logit model, we obtain

```

. * AME after logit
. qui logit ins i.retire age i.hstatusg hhincome educyear i.married i.hisp,
>      vce(robust)
. margins, dydx(*) noatlegend          // (AME)
Average marginal effects                                         Number of obs = 3,206
Model VCE: Robust
Expression: Pr(ins), predict()
dy/dx wrt: 1.retire age 1.hstatusg hhincome educyear 1.married 1.hisp

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.retire	.0426943	.0183334	2.33	0.020	.0067615	.0786272
age	-.0031693	.0023904	-1.33	0.185	-.0078543	.0015158
1.hstatusg	.0675283	.0197001	3.43	0.001	.0289167	.1061399
hhincome	.0005002	.000248	2.02	0.044	.0000141	.0009863
educyear	.0248111	.0030334	8.18	0.000	.0188658	.0307565
1.married	.1235562	.0194318	6.36	0.000	.0854706	.1616418
1.hisp	-.1608825	.0336124	-4.79	0.000	-.2267616	-.0950034

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

For example, on average across individuals, retirement is associated with a 0.0427 higher probability of having insurance.

The AMEs for the regressors are approximately 0.2 times the logit coefficient estimates. And these logit model AMEs are similar to the coefficients from OLS regression given in section [17.4.4](#). This is the usual case for single-index models such as logit.

The AME just computed is an unweighted sample average. To obtain a population AME, we should additionally use the *weight* modifier of the `margins` command.

17.6.2 MEM

An alternative is to compute the ME for the average individual, using the `atmeans` option of the `margins, dydx()` command. We obtain

```

. * MEM after logit
. qui logit ins i.retire age i.hstatusg hhincome educyear i.married i.hisp,
>      vce(robust)
. margins, dydx(*) atmeans noatlegend // (MEM)
Conditional marginal effects                                         Number of obs = 3,206
Model VCE: Robust
Expression: Pr(ins), predict()
dy/dx wrt: 1.retire age 1.hstatusg hhincome educyear 1.married 1.hisp

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.retire	.0457255	.0195761	2.34	0.020	.0073571	.0840939
age	-.0034129	.0025767	-1.32	0.185	-.0084631	.0016373
1.hstatusg	.0716613	.0206194	3.48	0.001	.031248	.1120745
hhincome	.0005386	.0002692	2.00	0.045	.000011	.0010663
educyear	.0267179	.0033438	7.99	0.000	.0201642	.0332716
1.married	.1295601	.0199356	6.50	0.000	.090487	.1686332
1.hisp	-.1677028	.0338153	-4.96	0.000	-.2339796	-.1014261

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

For example, for the average individual, retirement is associated with a 0.0457 higher probability of having insurance. In this particular case, the MEMS are 0–10% greater than the AMEs.

17.6.3 MER

At times, it is useful to obtain the ME for a representative individual also. Note that the preceding MEMs were computed at sample average values, such as at `married` = 0.733, which literally means 73.3% married. The `at()` option of the `margins, dydx()` command computes the MER.

We use as a benchmark a 75-year-old, retired, married Hispanic with good health status, 12 years of education, and an income equal to 35. Then,

```

. * MER after logit
. qui logit ins i.retire age i.hstatusg hhincome educyear i.married i.hisp,
>      vce(robust)
. margins, dydx(*) at (retire=1 age=75 hstatusg=1 hhincome=35 educyear=12
>      married=1 hisp=1) noatlegend // (MER)
Conditional marginal effects                                         Number of obs = 3,206
Model VCE: Robust
Expression: Pr(ins), predict()
dy/dx wrt: 1.retire age 1.hstatusg hhincome educyear 1.married 1.hisp

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.retire	.0354151	.0150517	2.35	0.019	.0059142	.0649159
age	-.0027608	.0020208	-1.37	0.172	-.0067216	.0012
1.hstatusg	.0544316	.016298	3.34	0.001	.022488	.0863752
hhincome	.0004357	.0002199	1.98	0.048	4.70e-06	.0008668
educyear	.0216131	.0036858	5.86	0.000	.014389	.0288372
1.married	.0935092	.0172199	5.43	0.000	.0597587	.1272597
1.hisp	-.1794232	.0378334	-4.74	0.000	-.2535752	-.1052712

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

For example, for this particular individual, retirement is associated with a 0.0354 higher probability of having insurance. The MERS for the regressors are approximately 20% less than the AMES.

17.6.4 The prchange command for MEs

The community-contributed `prchange` command ([Long and Freese 2014](#)) supplements the ME calculation that can be obtained using the `margins` command by reporting changes in probability induced by several types of change in the regressor, and not just a unit change or an infinitesimal change in the regressor.

The syntax is similar to that of `prvalue`, discussed in section [17.5.7](#),

`prchange [varname] [if] [in] [, x(conditions) rest(stat) options]`

where `varname` is the variable that changes. The default for the conditioning variables is the sample mean.

The following gives the ME of a change in income (`hhincome`) evaluated at the mean of regressors evaluated at $\mathbf{x} = \bar{\mathbf{x}}$.

```
. * Computing change in probability after logit
. qui logit ins retire $xlist, vce(robust)
. prchange hhincome

logit: Changes in Probabilities for ins

      min->max      0->1      -+1/2      -+sd/2  MargEfct
hhincome    0.5679    0.0005    0.0005    0.0346    0.0005

          0          1
Pr(y|x)  0.6272  0.3728

      retire      age  hstatusg  hhincome  educyear  married      hisp
  x=   .624766   66.9139   .704616   45.2639   11.8986   .733001   .072676
sd_x=   .484259   3.67579   .456286   64.3394   3.30461   .442461   .259645
```

The output `min->max` gives the change in probability due to income changing from the minimum to the maximum observed value. The output `0->1` gives the change due to income changing from 0 to 1. The output `-+1/2` gives the impact of income changing from a half unit below to a half unit above the base value. And the output `-+sd/2` gives the impact of income changing from one-half a standard deviation below to one-half a standard deviation above the base value. The final column gives the MEM and equals the result obtained earlier using the `margins, dydx(*) atmeans` command. Adding the `help` option to this command generates explanatory notes for the computer output.

17.7 Clustered data

Section 13.9 presented in some detail various models and methods that can be applied when observations in the same cluster are correlated while observations in different clusters are uncorrelated. That discussion was illustrated using the Poisson model. Here we provide a brief summary of adaptation to logit and probit models.

The simplest approach is to continue to use the `logit` and `probit` commands but use the `vce(cluster clustvar)` option. This maintains the assumption that the probability function for individual i in cluster g is $\Pr(y_{ig}|\mathbf{x}_{ig}) = F(\mathbf{x}'_{ig}\boldsymbol{\beta})$. But it provides corrected standard errors that adjust for the loss in precision that arises because of observations no longer being independent within cluster. As with any cluster-robust estimate of standard errors, we assume that there are many clusters.

Potentially more efficient estimates can be obtained by estimating using nonlinear feasible generalized least squares, assuming equicorrelation within cluster. For the logit model, this population-averaged approach uses the `xtgee` command with options `family(binomial)`, `link(logit)`, and `corr(exchangeable)`. It is good practice to add the `vce(robust)` option because this provides cluster-robust standard errors that guard against within-cluster correlation not being exactly one of equicorrelation. An equivalent command is `xtlogit, pa corr(exchangeable) vce(robust)`. One must first use the `xtset` command to define the cluster variable.

Another way to obtain potentially more efficient estimates is to introduce a random intercept, so $\Pr(y_{ig}|\mathbf{x}_{ig}, \alpha_g) = F(\alpha_g + \mathbf{x}'_{ig}\boldsymbol{\beta})$, where α_g is normally distributed. For the logit model, for example, one uses the `xtlogit, re` command. Equivalently, one can use the `melogit` command, which has the option of additionally allowing random-slope coefficients. Estimator consistency in this `re` model requires that the random effects be independent and identically distributed $N(\alpha, \sigma^2_\alpha)$. So even though there is a `vce(robust)` option, if it needs to be used, then the parameter estimates are most likely inconsistent.

Fixed-effects estimation is possible when there are many observations within each cluster. For example, in a regular logit or probit regression, simply add `i.cid` as regressors, where `cid` denotes the cluster identifier variable.

When there are few observations per cluster, consistent estimation of a fixed-effects model is generally no longer possible because of the incidental parameters problem. It is possible for the logit model, using the `xtlogit, fe` command, to fit a conditional logit model, though it is not possible for the probit model. An alternative is the correlated random-effects model that fits a random-effects model with cluster-means of regressors as additional regressors; see section 6.6.5 for the linear case and section [22.4.9](#) for a logit panel example.

Recent research has proposed bias-corrected methods for logit and probit estimation with fixed effects introduced as dummy variables for the case where there is a moderate number of clusters. These methods are implemented using the community-contributed commands `probitfe` and `logitfe` for probit and logit models; see [Cruz-Gonzalez, Fernández-Val, and Weidner \(2017\)](#) and section [22.4.8](#).

For proportions data, such as the share of an individual's budget devoted to food, with the additional complication of clustering, the methods given in section [17.3.5](#) for the `glm` command can be adapted to the `xtgee` command. For the logit model, one uses the `xtgee` command with options `link(logit)`, `corr(exchangeable)`, and `vce(robust)`.

The same issues arise with panel data. In that case, each individual is a cluster, and the observations for a given cluster are data for each time period that the individual is observed. A lengthier discussion for panel data on binary outcomes is given in section [22.4](#).

17.8 Additional models

We present models for binary outcomes that specify a more flexible functional form for the probability than the logit or probit models.

17.8.1 Heteroskedastic probit model

The standard probit and logit models can be motivated using the latent-variable model (17.3). This model assumes homoskedasticity of the errors, u , in the latent-variable model (17.3), a restriction that can be relaxed.

Letting u_i in (17.3) be heteroskedastic with a variance of

$$\sigma_i^2 = \exp(\mathbf{z}'_i \boldsymbol{\delta}) \quad (17.6)$$

we obtain the heteroskedastic probit model

$$\Pr(y_i = 1 | \mathbf{x}) = \Phi(\mathbf{x}'_i \boldsymbol{\beta} / \sigma_i) \quad (17.7)$$

where the exogenous variables (z_1, \dots, z_m) do not contain a constant, so that the restriction $\boldsymbol{\delta} = \mathbf{0}$ yields $\sigma_i^2 = 1$ as in the standard probit model (including a constant in \mathbf{z} would make the model unidentified).

Note that the novelty is allowing the latent variable y_i^* to be heteroskedastic. The observed binary variable y_i is necessarily heteroskedastic even in the simpler probit model.

ML estimation can be based on (17.6) and (17.7). The parameters of the probit model with heteroskedasticity can be estimated with ML by using Stata's `hetprobit` command. The syntax for `hetprobit` is

```
hetprobit depvar [indepvars] [if] [in] [weight],  
    het(varlist [, offset(varname_o)]) [options]
```

As an illustration, we extend the probit model used in the preceding analysis, with \mathbf{z} composed of the two variables `age`, already included in the \mathbf{x} variable, and `chronic`. We obtain

```

. * Heteroskedastic probit model
. hetprobit ins retire $xlist, het(age chronic) nolog vce(robust)

Heteroskedastic probit model                               Number of obs     =      3,206
                                                               Zero outcomes    =      1,965
                                                               Nonzero outcomes =      1,241
                                                               Wald chi2(7)    =       0.96
Log pseudolikelihood = -1992.804                      Prob > chi2    =   0.9954

```

		Robust				
	ins	Coefficient	std. err.	z	P> z	[95% conf. interval]
ins	retire	.1887091	.2547117	0.74	0.459	-.3105166 .6879348
	age	-.0190353	.0329883	-0.58	0.564	-.0836911 .0456205
	hstatusg	.2877448	.3768167	0.76	0.445	-.4508024 1.026292
	hhincome	.0020191	.0030096	0.67	0.502	-.0038796 .0079178
	educyear	.1142743	.1485502	0.77	0.442	-.1768787 .4054272
	married	.5898016	.7571601	0.78	0.436	-.8942049 2.073808
	hisp	-.7520952	.9298066	-0.81	0.419	-2.574483 1.070292
	_cons	-1.380543	1.531981	-0.90	0.368	-4.38317 1.622083
lnsigma	age	.0084299	.0189439	0.44	0.656	-.0286993 .0455592
	chronic	-.0406405	.0364957	-1.11	0.265	-.1121707 .0308898

Wald test of lnsigma=0: chi2(2) = 1.80 Prob > chi2 = 0.4071

The two models can be compared by using a Wald test of $\delta = 0$ that is automatically implemented when the command is used. The Wald test indicates that at the 0.05 level, there is no statistically significant improvement in the model resulting from generalizing the homoskedastic model because $p = 0.41$.

As a matter of modeling strategy, however, it is better to test first whether the z variables are omitted explanatory variables from the conditional mean model because such a misspecification is also consistent with variance depending on z . That is, the finding that z enters the variance function is also consistent with it having been incorrectly omitted from the conditional mean function. Accordingly, a variable addition test was also applied by adding `chronic` to the regressors in the probit model, and the p -value of the test was

found to be 0.23. Thus, the evidence is also against the inclusion of `chronic` in the probit model.

Parameter interpretation is more complicated in the heteroskedastic probit model, especially for a variable that appears in both \mathbf{x} and \mathbf{z} . The `margins` command provides `MES` that control for this complication.

17.8.2 Generalized logit

[Stukel \(1988\)](#) considered, as an alternative to the logit model, the generalized h -family logit model

$$\Lambda_\alpha(\mathbf{x}'\boldsymbol{\beta}) = \frac{e^{h_\alpha(\mathbf{x}'\boldsymbol{\beta})}}{1 + e^{h_\alpha(\mathbf{x}'\boldsymbol{\beta})}} \quad (17.8)$$

where $h_\alpha(\mathbf{x}'\boldsymbol{\beta})$ is a strictly increasing nonlinear function of $\mathbf{x}'\boldsymbol{\beta}$ indexed by two shape parameters α_1 and α_2 that govern, respectively, the heaviness of the tails and the symmetry of the $\Lambda(\cdot)$ function.

Rather than fit this richer model, Stukel proposed testing whether (17.8) is a better model by using a Lagrange multiplier, or score, test; see section 11.5. This test has the advantage that it requires estimation only of the null hypothesis logit model rather than of the more complicated model (17.8). Furthermore, the Lagrange multiplier test can be implemented by supplementing the logit model regressors with generated regressors that are functions of $\mathbf{x}'\boldsymbol{\beta}$ and by testing the significance of these augmented regressors.

For example, to test for departure from the logit in the direction of an asymmetric h -family, we add the generated regressor $(\mathbf{x}'_i\hat{\boldsymbol{\beta}})^2$ to the list of regressors, refit the logit model, and test whether the added variable significantly improves the fit of the model. We have

```

. * Stukel score or Lagrange multiplier test for asymmetric h-family logit
. qui logit ins retire $xlist
. predict xbhat, xb
. generate xbhatsq = xbhat^2
. qui logit ins retire $xlist xbhatsq, vce(robust)
. test xbhatsq
( 1) [ins]xbhatsq = 0
      chi2( 1) =    28.84
      Prob > chi2 =    0.0000

```

The null hypothesis of correct model specification is strongly rejected because the Wald test of zero coefficient for the added regressor $(\mathbf{x}_i' \hat{\beta})^2$ yields a $\chi^2(1)$ statistic of 28.8 with $p = 0.000$.

This test is easy to apply and so are several other score tests suggested by Stukel that use the variable-augmentation approach. At the same time, recall from section 3.7.6 that tests have power in more than one direction. Thus, rejection in the previous example may be for reasons other than the need for an asymmetric h -family logit model. For example, perhaps it is enough to use a logit model with additional inclusion of polynomials in the continuous regressors or inclusion of additional variables as regressors.

17.8.3 Alternative-specific random parameters logit or mixed logit

Binary outcome data are often the result of choice between two alternatives. Let the two alternatives be denoted 0 and 1, and suppose that some regressors vary at both the individual level and the alternative level, while other regressors vary over the individual level. Specifically, let \mathbf{x}_{i0} and \mathbf{x}_{i1} denote values of regressors that vary across the two alternatives, and let \mathbf{z}_i denote regressors that do not vary across alternatives. A simple form of this model can be fit by logit regression of y_i on $\mathbf{x}_{i1} - \mathbf{x}_{i0}$ and \mathbf{z}_i , where $y_i = 1$ if alternative 1 is chosen.

The alternative-specific random parameters logit model or mixed model specifies the alternative-varying regressors \mathbf{x}_{i0} and \mathbf{x}_{i1} to have associated parameters β_i that vary at the individual level as independent draws from a normal distribution. This model is used much more with multinomial outcomes, rather than binary outcomes, and is presented in detail in section [18.8](#).

Note that the model just described is a random parameters or mixed model different from the models in sections 13.9.3 and [17.7](#), which have intercept and possibly slope coefficients varying by cluster. Here the variation is by individual for alternative-specific regressors.

17.8.4 Nonparametric logit estimation

The focus of this chapter has been obtaining parameter estimates and associated MES for binary outcomes.

We may instead want to predict the outcome itself, for example, whether an individual has private insurance ($\hat{y} = 1$) or does not have insurance ($\hat{y} = 0$). Methods specifically intended for classification that are more flexible than logit and probit, such as discriminant analysis and support vector machines, are summarized in section [28.6.7](#).

And we may want to predict the probability that $y = 1$. This is an essential input for propensity-score matching methods and inverse-probability weighted estimators that are extensively used in the treatment-effects literature. One way to proceed is to fit a logit model with a flexible combination of regressors that includes interactions and powers of key variables. An alternative method fits a nonparametric model, albeit one that restricts the probability to lie between 0 and 1.

We consider the local logit estimator at $\mathbf{x} = \mathbf{x}_0$ that maximizes with respect to α_0 and β_0 the weighted logit log density

$$\sum_{i=1}^N w_h(\mathbf{x}_i - \mathbf{x}_0) \times (y_i \ln \Lambda \{\alpha_0 + (\mathbf{x}_i - \mathbf{x}_0)' \beta_0\} + (1 - y_i) \ln [1 - \Lambda \{\alpha_0 + (\mathbf{x}_i - \mathbf{x}_0)' \beta_0\}])$$

where $w_h(\mathbf{x}_i - \mathbf{x}_0)$ are kernel weights. This is the same as the local linear estimator presented in section [27.2](#), except that squared residuals are replaced by the log density of the logit model.

The community-contributed `ivqte` package (see section [25.9](#)) includes a command `locreg` that with option `logit` implements the local logit estimator.

We apply this command to the example of this chapter. For continuous regressors, we use the default `epan2` with bandwidth in the range (0.5, 0.8), and for binary regressors, we use the Li–Racine kernel with bandwidth in the range (0.8, 1). We obtain

```
. * Local logit regression
. locreg ins, dummy(retire hstatusg married hisp)
>     continuous(age hhincome educyear) logit bandwidth(0.5 0.8)
>     lambda(0.8 1.0) generate(ploclog, replace)

Leave-one-out cross-validation
```

Bandwidth	Lambda	Mean Squared Error
.5	.8	.23070108
.5	1	.23062031
.8	.8	.21783086
.8	1	.21734733

Among the grid of values tested, the optimal bandwidth is .8 and the optimal
> lambda is 1.

The fitted values obtained with the optimal smoothing parameters have been
> saved in `ploclog`.

We obtain logit ML predictions and compare them with those from the local logit model.

```
. * Compare local logit and logit ML predicted probabilities
. qui logit ins retire $xlist
. qui predict plogitml
. qui scatter ploclog plogitml, xtitle("Logit ML predicted probability")
>     ytitle("Local logit predicted probability") mszie(tiny)
>     scale(1.2)
```

Figure 17.2 plots the predicted probabilities from the local logit model against those obtained following logit ML estimation. The local logit model provides a greater spread of predicted values, including some predictions at the boundary values of 0 and 1. In principle, this greater spread is preferred, though it will pose a challenge for inverse-probability weighting methods.

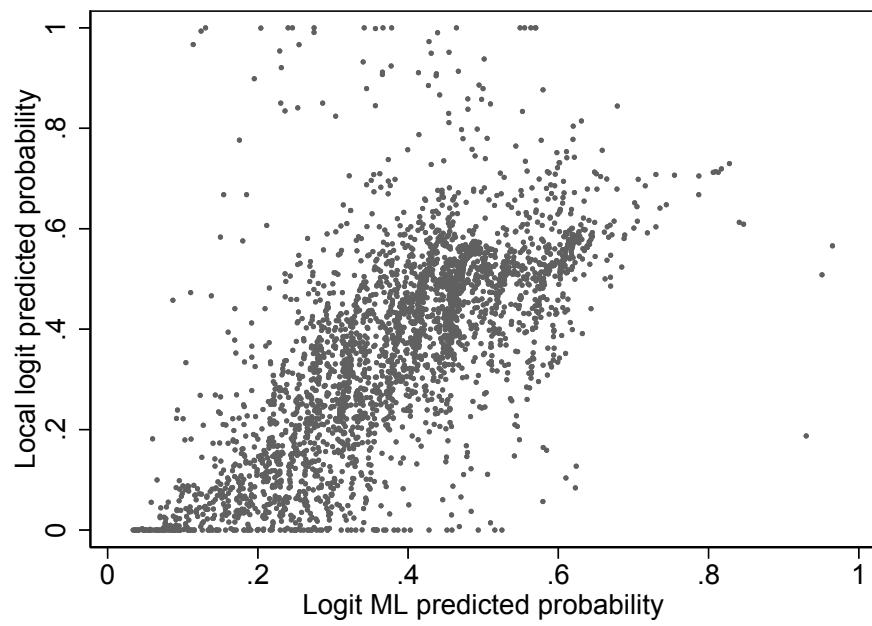


Figure 17.2. Predicted probabilities from local logit compared with logit ML

17.9 Endogenous regressors

The probit and logit ML estimators are inconsistent if any regressor is endogenous. Two distinct broad approaches are used to correct for endogeneity.

The structural approach specifies a complete model that explicitly models both nonlinearity and endogeneity. The specific structural model used differs according to whether the endogenous regressor is discrete or continuous. ML estimation is most efficient, but computationally simpler (albeit less efficient) two-step estimators are often used. [Wooldridge \(2010, chap. 15.7\)](#) provides a detailed exposition of these methods.

The alternative partial model or semiparametric approach defines an error term for the equation of interest and uses the IV estimator based on the orthogonality of instruments and this error term.

As in the linear case, a key requirement is the existence of one or more valid instruments that do not directly explain the binary dependent variable but are correlated with the endogenous regressor. Unlike the linear case, different approaches to controlling for endogeneity can lead to different estimators even in the limit because the parameters of different models are being estimated.

17.9.1 Example

We again model the binary outcome `ins`, though we use a different set of regressors. The regressors include the continuous variable `linc` (the log of household income) that is potentially endogenous because purchase of supplementary health insurance and household income may be subject to correlated unobserved shocks, even after controlling for a variety of exogenous variables. That is, for the HRS sample under consideration, the choice of supplementary insurance (`ins`), as well as household income (`linc`), may be considered as jointly determined.

Regular probit regression that does not control for this potential endogeneity yields

```

. * Endogenous probit using inconsistent probit MLE
. generate linc = log(hhincome)
(9 missing values generated)
. global xlist2 female age age2 educyear married hisp white chronic adl hstatusg
. probit ins linc $xlist2, vce(robust) nolog
Probit regression
Number of obs = 3,197
Wald chi2(11) = 366.94
Prob > chi2 = 0.0000
Pseudo R2 = 0.0946
Log pseudolikelihood = -1933.4275

```

ins	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
linc	.3466893	.0402173	8.62	0.000	.2678648 .4255137
female	-.0815374	.0508549	-1.60	0.109	-.1812112 .0181364
age	.1162879	.1151924	1.01	0.313	-.109485 .3420608
age2	-.0009395	.0008568	-1.10	0.273	-.0026187 .0007397
educyear	.0464387	.0089917	5.16	0.000	.0288153 .0640622
married	.1044152	.0636879	1.64	0.101	-.0204108 .2292412
hisp	-.3977334	.1080935	-3.68	0.000	-.6095927 -.1858741
white	-.0418296	.0644391	-0.65	0.516	-.168128 .0844687
chronic	.0472903	.0186231	2.54	0.011	.0107897 .0837909
adl	-.0945039	.0353534	-2.67	0.008	-.1637953 -.0252125
hstatusg	.1138708	.0629071	1.81	0.070	-.0094248 .2371664
_cons	-5.744548	3.871615	-1.48	0.138	-13.33277 1.843677

The regressor `linc` has coefficient 0.35 and is quite precisely estimated with a standard error of 0.04. The associated ME at $\mathbf{x} = \bar{\mathbf{x}}$, computed using the `margins, dydx(linc) atmeans` command, is 0.13. This implies that for the average individual, a 10% increase in household income (a change of 0.1 in `linc`) is associated with an increase of 0.013 in the probability of having supplementary health insurance.

17.9.2 Structural model

We restrict attention to the case of a single continuous endogenous regressor in a binary outcome model. For a discrete endogenous regressor, other methods should be used.

We consider the following linear latent-variable model, in which y_1^* is the dependent variable in the structural equation and y_2 is a continuous endogenous regressor in this equation. These two endogenous variables are modeled as linear in exogenous variables \mathbf{x}_1 and \mathbf{x}_2 . That is,

$$y_{1i}^* = \beta y_{2i} + \mathbf{x}'_{1i} \gamma + u_i \quad (17.9)$$

$$y_{2i} = \mathbf{x}'_{1i} \pi_1 + \mathbf{x}'_{2i} \pi_2 + v_i \quad (17.10)$$

where $i = 1, \dots, N$; \mathbf{x}_1 is a $K_1 \times 1$ vector of exogenous regressors; and \mathbf{x}_2 is a $K_2 \times 1$ vector of additional instrumental variables that affect y_2 but can be excluded from (17.9) because they do not directly affect y_1 . Identification requires that $K_2 \geq 1$.

The variable y_1^* is latent and hence is not directly observed. Instead, the binary outcome y_1 is observed, with $y_1 = 1$ if $y_1^* > 0$ and $y_1 = 0$ if $y_1^* \leq 0$.

Equation (17.9) is interpreted as being “structural”; see section 7.2.1. System (17.9)–(17.10) has a special restricted triangular structure because (17.10) restricts y_{2i} to not depend on y_{1i}^* once \mathbf{x}_{1i} and \mathbf{x}_{2i} are included.

Equation (17.10), called a first-stage equation or reduced-form equation, serves only as a source of identifying instruments. It explains the variation in the endogenous variable in terms of strictly exogenous variables, including the instrumental variables \mathbf{x}_2 that are excluded from the structural equation. These excluded instruments, previously discussed in sections 7.2 and 7.3 within the context of linear models, are essential for identifying the parameters of the structural equation.

Given the specification of the structural and first-stage equations, estimation can be simultaneous (that is, joint) or sequential.

Before presenting these estimators, we note that an alternative model, a simultaneous-equations model, replaces the reduced-form equation (17.10) for y_{2i} with a structural equation that additionally includes y_{1i}^* as a regressor. The community-contributed `cdsimeq` command ([Keshk 2003](#)) implements a two-stage estimation method for that model.

17.9.3 Structural-model estimators

The structural-model approach completely specifies the distributions of y_1^* and y_2 in (17.9) and (17.10). It is assumed that (u_i, v_i) are jointly normally distributed, that is, $(u_i, v_i) \sim N(\mathbf{0}, \Sigma)$, where $\Sigma = (\sigma_{ij})$. In the binary probit model, the coefficients are identified up to a scale factor only; hence,

by scale normalization, $\sigma_{11} = 1$. The assumptions imply that $u_i|v_i = \rho v_i + \varepsilon_i$, where $E(\varepsilon_i|v_i) = 0$. A test of the null hypothesis of exogeneity of y_2 is equivalent to the test of $H_0: \rho = 0$ because then u_i and v_i are independent.

This approach relies greatly on the distributional assumptions. Consistent estimation requires both normality and homoskedasticity of the errors u_i and v_i .

The ivprobit command

The syntax of `ivprobit` is similar to that of `ivregress`, discussed in section 7.4.

```
ivprobit depvar [ varlist1 ] (varlist2=varlist_iv) [ if ] [ in ] [ weight ]
[ , mle_options ]
```

where `varlist2` refers to the endogenous variable y_2 and `varlist_iv` refers to the instruments \mathbf{x}_2 that are excluded from the equation for y_1^* . The default version of `ivprobit` delivers ML estimates, and the `twostep` option yields two-step estimates. The postestimation command `margins` following ML estimations provides marginal effects; see the *Remarks and examples* section of [`R`] **ivprobit postestimation**.

ML estimates

For this example, we use as instruments two excluded variables, `retire` and `sretire`. These refer to, respectively, individual retirement status and spouse retirement status. These are likely to be correlated with `linc` because retirement will lower household income. The key assumption for instrument validity is that retirement status does not directly affect choice of supplementary insurance. This assumption is debatable, and this example is best viewed as merely illustrative. We apply `ivprobit`, obtaining ML estimates:

```

. * Endogenous probit using ivprobit ML estimator
. global ivlist2 retire sretire
. ivprobit ins $xlist2 (linc = $ivlist2), vce(robust) nolog
Probit model with endogenous regressors                               Number of obs = 3,197
Log pseudolikelihood = -5407.7151                                     Wald chi2(11) = 382.35
                                                               Prob > chi2 = 0.0000

```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
linc	-.5338252	.3852132	-1.39	0.166	-.1288829 .2211788
female	-.1394072	.0494471	-2.82	0.005	-.2363218 -.0424926
age	.2862293	.1280821	2.23	0.025	.0351929 .5372656
age2	-.0021472	.0009318	-2.30	0.021	-.0039735 -.0003209
educyear	.1136881	.0237914	4.78	0.000	.0670579 .1603183
married	.7058309	.2377594	2.97	0.003	.239831 1.171831
hisp	-.5094514	.1049487	-4.85	0.000	-.715147 -.3037558
white	.1563454	.1035674	1.51	0.131	-.0466429 .3593338
chronic	.0061939	.027525	0.23	0.822	-.0477542 .060142
adl	-.1347664	.0349799	-3.85	0.000	-.2033258 -.0662071
hstatusg	.2341789	.0709755	3.30	0.001	.0950694 .3732883
_cons	-10.00787	4.065771	-2.46	0.014	-17.97664 -2.039107
corr(e.linc, e.ins)	.5879559	.2355329			-.0309872 .8809669
sd(e.linc)	.7177787	.0167816			.6856296 .7514352

Wald test of exogeneity (corr = 0): chi2(1) = 3.51 Prob > chi2 = 0.0610
 Instrumented: linc
 Instruments: female age age2 educyear married hisp white chronic adl
 hstatusg retire sretire

The output includes a test of the null hypothesis of exogeneity, that is, $H_0: \rho = 0$. The p -value is 0.061, so H_0 is not rejected at the 0.05 level, though it is rejected at the 0.10 level. That the estimated coefficient is positive indicates a positive correlation between u and v . Those unmeasured factors that make it more likely for an individual to have a higher household income also make it more likely that the individual will have supplementary health insurance, conditional on other regressors included in the equation.

Given the large estimated value for ρ ($\hat{\rho} = 0.59$), we should expect that the coefficients of the estimated `probit` and `ivprobit` models differ. This is indeed the case, for both the endogenous regressor `linc` and for the other regressors. The coefficient of `linc` actually changes signs (from 0.35 to -0.53), so that an increase in household income is estimated to lower the

probability of having supplementary insurance. One possible explanation is that richer people are willing to self-insure for medical services not covered by Medicare. At the same time, IV estimation has led to much greater imprecision, with the standard error increasing from 0.04 to 0.39, so that the negative coefficient is not statistically significantly different from 0 at the 0.05 level. Taken at face value, however, the result suggests that the `probit` command that neglects endogeneity leads to an overestimate of the effect of household income. The remaining coefficients exhibit the same sign pattern as in the ordinary probit model, and the differences in the point estimates are within the range of estimated standard errors.

The same ML estimates are obtained using the following `eprobit` command:

```
. * Endogenous probit using eprobit ML estimator (an ERM command)
. eprobit ins $xlist2, endogenous(linc = $xlist2 $ivlist2) vce(robust) nolog
(output omitted)
```

The suppressed output includes estimates of the first-stage equation for `linc`. To obtain this additional output using the `ivprobit` command, one adds the `first` option.

The `eprobit` command is a member of the class of extended regression model commands that provide ML estimates for models with endogenous regressors or sample selection, or both, when the full model is a recursive model such as (17.9) and (17.10) and error terms are joint normal distributed. Section 23.7 provides an overview of extended regression model commands and several examples of the `eprobit` command.

Control function estimator

Rivers and Vuong (1988) proposed a control function estimator analogous to that presented in section 7.4.7. The residual \hat{v}_i from first-stage OLS regression of (17.10) is included as an additional regressor in a second-stage probit regression of (17.9). A test of whether the coefficient of this extra term is zero is a test of endogeneity. If there is endogeneity, then adjustment is needed because inference needs to account for first-stage estimation to compute \hat{v}_i and because parameter estimates need to be

rescaled by $\sqrt{1 - \rho^2}$, where $\rho = \text{Cor}(u_i, v_i)$. One can bootstrap or adapt the two-step method of section 13.3.11.

This two-step estimator requires the same strong distributional assumptions as the MLE. The advantage is computational, especially for extension to more than one endogenous regressor.

Two-step sequential estimates

[Newey \(1987\)](#) proposed an alternative two-step estimator, a minimum chi-squared estimator, that also is computationally simpler than the MLE but is more efficient than the preceding two-step estimator of [Rivers and Vuong \(1988\)](#). It also requires the same strong distributional assumptions as the MLE.

The estimator is implemented by using `ivprobit` with the `twostep` option.

We do so for our data, using the `first` option, which also provides the least-squares (OLS) estimates of the first stage. Note that robust standard errors are unavailable.

. * Endogenous probit using ivprobit two-step estimator
 . ivprobit ins \$xlist2 (linc = \$ivlist2), twostep first
 Checking reduced-form model...
 first-stage regression

Source	SS	df	MS	Number of obs	=	3,197
Model	1173.12053	12	97.7600445	F(12, 3184)	=	188.99
Residual	1647.03826	3,184	.517285885	Prob > F	=	0.0000
				R-squared	=	0.4160
Total	2820.15879	3,196	.882402626	Adj R-squared	=	0.4138
				Root MSE	=	.71923

linc	Coefficient	Std. err.	t	P> t	[95% conf. interval]
retire	-.0909581	.0288119	-3.16	0.002	-.1474499 -.0344663
sretire	-.0443106	.0317252	-1.40	0.163	-.1065145 .0178932
female	-.0936494	.0297304	-3.15	0.002	-.151942 -.0353569
age	.2669284	.0627794	4.25	0.000	.1438361 .3900206
age2	-.0019065	.0004648	-4.10	0.000	-.0028178 -.0009952
educyear	.094801	.0043535	21.78	0.000	.0862651 .1033369
married	.7918411	.0367275	21.56	0.000	.7198291 .8638531
hisp	-.2372014	.0523874	-4.53	0.000	-.3399179 -.134485
white	.2324672	.0347744	6.69	0.000	.1642847 .3006496
chronic	-.0388345	.0100852	-3.85	0.000	-.0586086 -.0190604
adl	-.0739895	.0173458	-4.27	0.000	-.1079995 -.0399795
hstatusg	.1748137	.0338519	5.16	0.000	.10844 .2411875
_cons	-7.702456	2.118657	-3.64	0.000	-11.85653 -3.548385

Two-step probit with endogenous regressors

Number of obs	=	3,197
Wald chi2(11)	=	222.51
Prob > chi2	=	0.0000

	Coefficient	Std. err.	z	P> z	[95% conf. interval]
linc	-.6109088	.5723054	-1.07	0.286	-1.732607 .5107893
female	-.167917	.0773839	-2.17	0.030	-.3195867 -.0162473
age	.3422526	.1915485	1.79	0.074	-.0331756 .7176808
age2	-.0025708	.0014021	-1.83	0.067	-.0053188 .0001773
educyear	.13596	.0543047	2.50	0.012	.0295249 .2423952
married	.8351517	.441743	1.89	0.059	-.0306487 1.700952
hisp	-.6184546	.181427	-3.41	0.001	-.9740451 -.2628642
white	.1818279	.1528281	1.19	0.234	-.1177098 .4813655
chronic	.0095837	.0309618	0.31	0.757	-.0511004 .0702678
adl	-.1630884	.0568288	-2.87	0.004	-.2744709 -.0517059
hstatusg	.2809463	.1228386	2.29	0.022	.0401871 .5217055
_cons	-12.04848	5.928158	-2.03	0.042	-23.66746 -.4295071

Wald test of exogeneity: chi2(1) = 3.57

Prob > chi2 = 0.0588

Instrumented: linc

Instruments: female age age2 educyear married hisp white chronic adl
 hstatusg retire sretire

The results of the two-step estimator are similar to those from the `ivprobit` ML estimation. The coefficient estimates are within 20% of each other. The standard errors are increased by approximately 50%, indicating a loss of precision in two-step estimation compared with ML estimation. The test statistic for exogeneity of `linc` has a *p*-value of 0.059 compared with 0.061 using ML. The results for the first stage indicate that one of the two excluded instrumental variables has a strong predictive value for `linc`. Because this is a reduced-form equation, we do not attempt an interpretation of the results.

Weak instruments

If instruments are weakly correlated with the endogenous regressor, then the usual asymptotic theory may perform poorly. Then alternative methods may be used. Inference for weak instruments for the linear model is presented in section 7.7.

The discussion there includes methods based on minimum distance estimation due to [Magnusson \(2010\)](#) that can be applied to a wide range of linear and nonlinear structural models. The community-contributed `rivtest` ([Finlay and Magnusson 2009](#)) and `weakiv` programs ([Finlay, Magnusson, and Schaffer 2014](#)) apply these methods following estimation using `ivprobit`.

17.9.4 Linear two-stage least-squares approach

A popular approach is to ignore the binary nature of the dependent variable y_1 (`ins`) and simply estimate by two-stage least-squares (2SLS).

Advantages of the linear 2SLS estimator are its computational simplicity and the ability to use tests of validity of overidentifying instruments and diagnostics for weak instruments in linear models that were presented in chapter 7. At the same time, formal tests and inference that require normal homoskedastic errors will be inappropriate because of the intrinsic heteroskedasticity when the dependent variable is binary.

Our own view is that at best linear 2SLS may provide an initial guide, but ultimately one should use other methods that allow for the binary nature of the dependent variable.

We have the standard linear formulation for the observed variables (y_1, y_2) ,

$$y_{1i} = \beta y_{2i} + \mathbf{x}'_{1i} \gamma + u_i$$

$$y_{2i} = \mathbf{x}'_{1i} \pi_1 + \mathbf{x}'_{2i} \pi_2 + v_i$$

where y_2 is endogenous and the covariates \mathbf{x}_2 are the excluded exogenous regressors (instruments). This is the model (17.9) and (17.10), except that the latent-variable y_1^* is replaced by the binary variable y_1 . An important difference is that while (u, v) are zero-mean and jointly dependent, they need not be multivariate normal and homoskedastic.

Estimation is by 2SLS, using the `ivregress` command. Because y_1 is binary, the error u is heteroskedastic. The 2SLS estimator is then still consistent for (β, γ) , but heteroskedasticity-robust standard errors should be used for inference.

The `ivregress` command with the `vce(robust)` option yields

```
. * Endogenous probit using ivregress to get 2SLS estimator
. ivregress 2sls ins $xlist2 (linc = $ivlist2), vce(robust) noheader
```

ins	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
linc	-.167901	.1937801	-0.87	0.386	-.547703	.2119011
female	-.0545806	.0260643	-2.09	0.036	-.1056657	-.0034955
age	.106631	.0624328	1.71	0.088	-.015735	.228997
age2	-.0008054	.0004552	-1.77	0.077	-.0016977	.0000868
educyear	.0416443	.0182207	2.29	0.022	.0059324	.0773562
married	.2511613	.1499264	1.68	0.094	-.042689	.5450116
hisp	-.154928	.0546479	-2.84	0.005	-.2620358	-.0478202
white	.0513327	.0508817	1.01	0.313	-.0483936	.151059
chronic	.0048689	.0103797	0.47	0.639	-.015475	.0252128
adl	-.0450901	.0174479	-2.58	0.010	-.0792874	-.0108928
hstatusg	.0858946	.041327	2.08	0.038	.0048951	.1668941
_cons	-3.303902	1.920872	-1.72	0.085	-7.068743	.4609388

Instrumented: linc

Instruments: female age age2 educyear married hisp white chronic adl
hstatusg retire sretire

```
. estat overid
```

Test of overidentifying restrictions:

```
Score chi2(1) = .521843 (p = 0.4701)
```

This method yields a coefficient estimate of -0.17 of `linc` that is statistically insignificant at level 0.05, as for `ivprobit`. To compare `ivregress` estimates with `ivprobit` estimates, we need to rescale parameters as in section [17.4.4](#). Then the rescaled 2SLS parameter estimate is $-0.17 \times 2.5 = -0.42$, comparable with the estimates of -0.53 and -0.61 from the `ivprobit` command.

Here the single overidentifying restriction is not rejected by the Hansen J test, which yields a $\chi^2(1)$ value of 0.522.

17.9.5 Nonlinear IV approach

If an IV estimation approach is taken, it is better to use the following nonlinear IV estimator that adapts for the binary nature of the dependent variable.

The linear 2SLS estimator of the model [\(17.11\)](#) is based on the moment condition $E(u|\mathbf{x}_1, \mathbf{x}_2) = 0$, where $u = y_1 - (\beta y_2 + \mathbf{x}'_1 \gamma)$; see section 7.3.2. For a binary outcome y_1 modeled using the probit model, it is better to instead define the error term, the difference between y_1 and its conditional mean function, as $u = y_1 - \Phi(\beta y_2 + \mathbf{x}'_1 \gamma)$.

There is no Stata command to implement the subsequent nonlinear IV estimator, but the nonlinear IV example in section 13.3.10 for the Poisson can be suitably adapted, replacing $\exp(\mathbf{x}'_i \beta)$ with $\Phi(\mathbf{x}'_i \beta)$ for probit or $\Lambda(\mathbf{x}'_i \beta)$ for logit. Estimation is based on a moment condition that is not implied by [\(17.9\)](#) and [\(17.10\)](#), so the estimators will differ even in the limit from those from the `ivprobit` command.

For the current overidentified example, two-step nonlinear IV or generalized method of moments estimates are obtained using the command

```
. * Endogenous probit using nonlinear IV or generalized method of
> moments estimation
. gmm (ins - normal({xb:linc $xlist2 _cons})), instruments($xlist2 $ivlist2)
  (output omitted)
```

From output not given, the endogenous regressor `linc` has coefficient -0.490 , compared with -0.534 and -0.611 from `ivprobit`, with robust

standard error 0.568. The postestimation command `estat overid` performs an overidentifying restrictions test.

17.10 Grouped and fractional data

In some applications, only grouped or aggregate data may be available, yet individual behavior is felt to be best modeled by a binary choice model. For example, we may have a frequency average taken across a sampled population as the dependent variable and averages of explanatory variables for the regressors, which we will assume to be exogenous. Such data are also called proportions or fractional data.

There are several ways to proceed, some introduced in section [17.3.5](#).

17.10.1 Grouped dataset

We use the dataset of this chapter with `age` as the grouping variable. This generates 33 groups, one for each age between 52 and 86; there are no observations for ages 84 or 85. The number of cases in the 33 groups are as follows:

4	5	2	2	7	8	34	62	72	51	61
67	74	524	470	488	477	286	133	100	91	67
36	29	19	11	8	11	4	6	5	1	1

Observations with no within-group variation are dropped; this is likely to occur when the group size is small. In the present sample, we drop two groups with 4 or fewer observations, reducing the sample size to 27. Because of the small sample size, we include only two variables in the regression models.

The full individual dataset of 3,206 observations can be converted to an aggregate dataset by using the following Stata commands that generate group averages, and counts the number of observations (and those with `ins==1`) in each cell.

```

. * Create grouped data
. qui use mus217hrs, clear
. bysort age: egen num_in_cell = count(ins)
. bysort age: gen num_ins_in_cell = num_in_cell*ins
. sort age
. collapse n=num_in_cell r=num_ins_in_cell av_ins=ins
>     av_educyear=educyear av_hstatusg=hstatusg, by(age)
. drop if n < 5
(6 observations deleted)
. summarize

```

Variable	Obs	Mean	Std. dev.	Min	Max
age	27	68	8.194745	53	82
n	27	118.2222	168.0651	5	524
r	27	45.88889	69.16165	1	226
av_ins	27	.3446255	.1068683	.125	.6666667
av_educyear	27	11.28696	1.0203	8.909091	12.5
av_hstatusg	27	.5789595	.2262814	0	.8888889

Here the `collapse` command with its default mean statistic is used to form averages by age. For example, `collapse av_ins=ins, by(age)` creates 27 observations for the `av_ins` variable equal to the average of the `ins` variable for each of the 27 distinct values taken by the `age` variable. More generally, `collapse` can compute other statistics, such as the median specifying the `median` statistic, and if the `by()` option was not used, then just a single observation would be produced.

The variable `n` gives the number of individuals in each distinct age group, and the variable `r` gives the number of individuals with insurance in each distinct age group, so $\text{avxins} = r/n$.

17.10.2 Comparison of various grouped estimators

We can view the data as coming from a binomial model with, for the g th group, r_g successes in n_g trials; here `r` successes in `n` trials. This model can be fit using the `glm` command with options `family(binomial)` and, for a logit model, `link(logit)`. We obtain

```

. * Grouped data: Binomial logit model for number insured (= n times y)
. glm r av_educyear av_hstatusg, family(binomial n) link(logit) noheader
Iteration 0:  log likelihood = -66.665299
Iteration 1:  log likelihood = -66.649602
Iteration 2:  log likelihood = -66.649602

```

r	OIM					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
av_educyear	.2187254	.1055729	2.07	0.038	.0118062	.4256445
av_hstatusg	.3309241	.4161667	0.80	0.427	-.4847477	1.146596
_cons	-3.296212	1.047531	-3.15	0.002	-5.349335	-1.24309

In the special case that, within each given age group, all individuals had the same value of the regressors, the preceding results would be identical to fitting a logit model on this grouped dataset expanded to have a separate observation for each individual.

The grouped data may be treated as proportions data, not necessarily arising from averaging individual binary outcomes. Then the proportion insured (av_ins) is modeled to have conditional mean $E(\bar{y}_g | \mathbf{x}) = \Lambda(\bar{\mathbf{x}}'_g \boldsymbol{\beta})$. Nonlinear least-squares estimation yields

```

. * Grouped data: NLS with actual y as dependent variable
. nl (av_ins = 1 / (1 + exp(-{xb: av_educyear av_hstatusg}+{b0}))), vce(robust)
> nolog

Nonlinear regression                               Number of obs =      27
                                                 R-squared =     0.9266
                                                 Adj R-squared =  0.9174
                                                 Root MSE =    .1035396
                                                 Res. dev. =   -49.01872

```

av_ins	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
/xb_av_educy~r	.1711613	.1173766	1.46	0.158	-.0710922	.4134148
/xb_av_hstat~g	.0763085	.4834848	0.16	0.876	-.921555	1.074172
/b0	2.625174	1.091327	2.41	0.024	.3727866	4.877561

An alternative proportions estimator, introduced in section [17.3.5](#), additionally models the heteroskedasticity. This uses the binary outcome objective function ([17.5](#)). The `logit` and `probit` commands cannot be applied to proportions data for the simple reason that they are coded to work only with a dependent variable that takes binary values. Instead, we use the

`fracreg logit` command (or `fracreg probit` for probit). Heteroskedastic-robust standard errors are computed by default; other options include cluster-robust standard errors. We obtain

```
. * Grouped data: "logit" with actual y as dependent variable
. fracreg logit av_ins av_educyear av_hstatusg, vce(robust) noheader nolog
```

av_ins	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
av_educyear	.1579758	.1045167	1.51	0.131	-.0468731	.3628247
av_hstatusg	.1147869	.4343325	0.26	0.792	-.7364891	.966063
_cons	-2.49697	.9749101	-2.56	0.010	-4.407759	-.5861815

In this example, there is some efficiency gain in modeling the heteroskedasticity, and this method is viewed as the best method for fractional response data.

A final approach uses the logistic transformation to redefine the dependent variable to be unbounded, rather than restricted to the $(0, 1)$ interval, and estimate by OLS the parameters of the model

$$\ln\left(\frac{\bar{y}_g}{1 - \bar{y}_g}\right) = \bar{\mathbf{x}}'_g \boldsymbol{\gamma} + u_g \quad (17.11)$$

where u_g is an error and robust standard errors are used. We obtain

```
. * Grouped data: least squares with transformation of y as dependent variable
. generate logins = log(av_ins/(1-av_ins))
. regress av_ins av_educyear av_hstatusg, vce(robust) noheader
```

av_ins	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
av_educyear	.0344108	.0231204	1.49	0.150	-.0133073	.0821289
av_hstatusg	.0287159	.0970939	0.30	0.770	-.1716761	.229108
_cons	-.0603931	.2151748	-0.28	0.781	-.504492	.3837058

A limitation of this method is that we are generally interested in explaining $E(\bar{y}_g | \mathbf{x}_g)$ rather than $E[\ln\{\bar{y}_g / (1 - \bar{y}_g)\} | \mathbf{x}_g]$.

17.11 Additional resources

The logit and probit models are the most commonly used nonlinear models, and for many purposes, the `logit` and `probit` commands are sufficient. For these nonlinear models, it is common to report the AME (obtained using the `margins, dydx()` command) in place of, or in addition to, parameter estimates. The most common complications for binary outcome models are endogenous regressors (see section [17.9](#)) and panel data (see section [22.4](#)).

17.12 Exercises

1. Consider the example of section [17.4](#) with dependent variable `ins` and the single regressor `educyear`. Estimate the parameters of logit, probit, and OLS models using both default and robust standard errors. For the regressor `educyear`, compare its coefficient across the models, compare default and robust standard errors of this coefficient, and compare the t statistics based on robust standard errors. For each model, compute the ME of one more year of education for someone with sample mean years of education, as well as the AME. Which model fits the data better—logit or probit?
2. Use the `cloglog` command to estimate the parameters of the binary probability model for `ins` with the same explanatory variables used in the `logit` model in this chapter. Estimate the AME for the regressors. Calculate the odds ratios of `ins=1` for the following values of the covariates: `age=50, retire=0, hstatusg=1, hhincome=45, educyear=12, married=1, and hisp=0`.
3. Generate a graph of fitted probabilities against years of education (`educyear`) or age (`age`) using as a template the commands used for generating figure [17.1](#) in this chapter.
4. Estimate the parameters of the logit model of section [17.4.2](#). Now, estimate the parameters of the probit model using the `probit` command. Use the reported log likelihoods to compare the models by the Akaike's information criterion and BIC.
5. Estimate the probit regression of section [17.4.4](#). Using the conditioning values (`age=65, retire=1, hstatusg=1, hhincome=60, educyear=17, married=1, hisp=0`), estimate and compare the ME of age on the $\Pr(\text{ins}=1 | \mathbf{x})$, using both the `margins` and `prchange` commands. They should give the same result.
6. Using the `hetprobit` command, estimate the parameters of the model of section [17.4](#), using `hhincome` as the sole variable determining the variance. Test the null hypothesis of homoskedastic probit.
7. Using the example in section [17.10](#) as a template, estimate a grouped logistic regression using `educyear` as the grouping variable. Comment on what you regard as unsatisfactory features of the grouping variable and the results.

Chapter 18

Multinomial models

18.1 Introduction

Categorical data are data on a dependent variable that can fall into one of several mutually exclusive categories. Examples include different ways to commute to work (by car, by bus, or on foot) and different categories of self-assessed health status (excellent, good, fair, or poor).

The econometrics literature focuses on modeling a single outcome from categories that are mutually exclusive, where the dependent variable outcome must be multinomial distributed, just as binary data must be Bernoulli or binomial distributed. Analysis is not straightforward, however, because there are many different models for the probabilities of the multinomial distribution. These models vary according to whether the categories are ordered or unordered, whether some of the individual-specific regressors vary across the alternative categories, and, in some settings, whether the model is consistent with utility maximization. Furthermore, parameter coefficients for any given model can be difficult to directly interpret. The marginal effects (MES) of interest measure the impact on the probability of observing each of several outcomes rather than the impact on a single conditional mean.

We begin with models for unordered outcomes, notably, multinomial logit (MNL), conditional logit (CL), nested logit (NL), multinomial probit (MNP), and alternative-specific random parameter logit models. We then move to models for ordered outcomes, such as health-status measures, and models for multivariate multinomial outcomes.

18.2 Multinomial models overview

We provide a general discussion of multinomial regression models. Subsequent sections detail the most commonly used multinomial regression models that correspond to particular functional forms for the probabilities of each alternative.

18.2.1 Probabilities and MEs

The outcome, y_i , for individual i is one of m alternatives. We set $y_i = j$ if the outcome is the j th alternative, $j = 1, 2, \dots, m$. The values $1, 2, \dots, m$ are arbitrary, and the same regression results are obtained if, for example, we use values $3, 5, 8, \dots$. The ordering of the values also does not matter, unless an ordered model (presented in section [18.9](#)) is used.

The probability that the outcome for individual i is alternative j , conditional on the regressors \mathbf{x}_i , is

$$p_{ij} = \Pr(y_i = j) = F_j(\mathbf{x}_i, \boldsymbol{\theta}), \quad j = 1, \dots, m, \quad i = 1, \dots, N \quad (18.1)$$

where different functional forms, $F_j(\cdot)$, correspond to different multinomial models. Only $m - 1$ of the probabilities can be freely specified because probabilities sum to one. For example, $F_m(\mathbf{x}_i, \boldsymbol{\theta}) = 1 - \sum_{j=1}^{m-1} F_j(\mathbf{x}_i, \boldsymbol{\theta})$. Multinomial models therefore require a normalization. Some Stata multinomial commands, including `cmclogit`, permit different individuals to face different choice sets so that, for example, an individual might be choosing only from among alternatives 1, 3, and 4.

The parameters of multinomial models are generally not directly interpretable. In particular, a positive coefficient need not mean that an increase in the regressor leads to an increase in the probability of an outcome being selected. Instead, we compute MES. For individual i , the ME of a change in the k th regressor on the probability that alternative j is the outcome is

$$\text{ME}_{ijk} = \frac{\partial \Pr(y_i = j)}{\partial x_{ik}} = \frac{\partial F_j(\mathbf{x}_i, \boldsymbol{\theta})}{\partial x_{ik}}$$

For each regressor, there will be m MES corresponding to the m probabilities, and these m MES sum to zero because probabilities sum to one. As for other nonlinear models, these MES vary with the evaluation point \mathbf{x} .

18.2.2 Maximum likelihood estimation

Estimation is by maximum likelihood (ML). We use a convenient form for the density that generalizes the method used for binary outcome models. The density for the i th individual is written as

$$f(y_i) = p_{i1}^{y_{i1}} \times \cdots \times p_{im}^{y_{im}} = \prod_{j=1}^m p_{ij}^{y_{ij}}$$

where y_{i1}, \dots, y_{im} are m indicator variables with $y_{ij} = 1$ if $y_i = j$ and $y_{ij} = 0$ otherwise. For each individual, exactly one of y_1, y_2, \dots, y_m will be nonzero. For example, if $y_i = 3$, then $y_{i3} = 1$, the other $y_{ij} = 0$, and upon simplification, $f(y_i) = p_{i3}$, as expected.

The likelihood function for a sample of N independent observations is the product of the N densities, so $L = \prod_{i=1}^N \prod_{j=1}^m p_{ij}^{y_{ij}}$. The maximum likelihood estimator (MLE), $\hat{\boldsymbol{\theta}}$, maximizes the log-likelihood function

$$\ln L(\boldsymbol{\theta}) = \sum_{i=1}^N \sum_{j=1}^m y_{ij} \ln F_j(\mathbf{x}_i, \boldsymbol{\theta}) \tag{18.2}$$

and as usual, default standard errors are based on $\hat{\boldsymbol{\theta}} \stackrel{a}{\sim} N(\boldsymbol{\theta}, [-E\{\partial^2 \ln L(\boldsymbol{\theta}) / \partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'\}]^{-1})$.

For multinomial models, the pseudo- R^2 has a meaningful interpretation; see section 13.8. Nonnested models can be compared by using the Akaike information criterion and related measures, assuming correct model specification.

For multinomial data, the key is specification of $F_j(\mathbf{x}_i, \boldsymbol{\theta})$. Various models for $F_j(\cdot)$ are presented in this chapter, with the suitability of a particular model depending on the application at hand.

18.2.3 Robust standard errors

For categorical data, the distribution is necessarily multinomial. Similar to the case for binary outcomes, in theory there is no advantage in using the robust sandwich form for the variance–covariance matrix of the estimator (VCE) of the MLE in the special case that data are independent over i and $F_j(\mathbf{x}_i, \boldsymbol{\theta})$ is correctly specified.

In practice, $F_j(\mathbf{x}_i, \boldsymbol{\theta})$ is likely to be misspecified. Then quasi-ML theory applies (see section 13.3.1), and we need to use robust standard errors that provide consistent estimates of the variance of $\hat{\boldsymbol{\theta}}$. So we generally obtain robust standard errors. For independent observations, we use the `vce(robust)` option, and for observations that are dependent because of clustering, we use the `vce(cluster clustvar)` option.

18.2.4 Case-specific and alternative-specific regressors

Some regressors, such as gender, do not vary across alternatives and are called case-specific or alternative-invariant regressors. Other regressors, such as price, may vary across alternatives and are called alternative-specific or case-varying regressors.

The commands used for multinomial model estimation can vary according to the form of the regressors. In the simplest case, all regressors are case specific, and we use the `mlogit` command.

In more complicated applications, some or all the regressors are alternative specific. Then one uses `cm` commands (choice-model commands)

introduced in Stata 16. For example, the `cmclogit` command supplants the `asclogit` command.

These different types of commands require data to be organized in different ways; see section [18.5.1](#).

18.2.5 Additive random-utility model

For unordered multinomial outcomes that arise from individual choice, econometricians favor models that come from utility maximization. This leads to multinomial models that are used much less in other branches of applied statistics.

For individual i and alternative j , we suppose that utility U_{ij} is the sum of a deterministic component, V_{ij} , that depends on regressors and unknown parameters and an unobserved random component ε_{ij} :

$$U_{ij} = V_{ij} + \varepsilon_{ij} \quad (18.3)$$

This is called an additive random-utility model (ARUM). We observe the outcome $y_i = j$ if alternative j has the highest utility of the alternatives. It follows that

$$\begin{aligned} \Pr(y_i = j) &= \Pr(U_{ij} \geq U_{ik}), \quad \text{for all } k \\ &= \Pr(U_{ik} - U_{ij} \leq 0), \quad \text{all } k \\ &= \Pr(\varepsilon_{ik} - \varepsilon_{ij} \leq V_{ij} - V_{ik}), \quad \text{all } k \end{aligned} \quad (18.4)$$

Standard multinomial models specify that $V_{ij} = \mathbf{x}'_i \boldsymbol{\beta} + \mathbf{z}'_i \boldsymbol{\gamma}_j$, where \mathbf{x}_i are alternative-specific regressors and \mathbf{z}_i are case-specific regressors. Different assumptions about the joint distribution of $\varepsilon_{i1}, \dots, \varepsilon_{im}$ lead to different multinomial models with different specifications for $F_j(\mathbf{x}_i, \boldsymbol{\theta})$ in [\(18.1\)](#). Because the outcome probabilities depend on the difference in errors, only $m - 1$ of the errors are free to vary, and similarly, only $m - 1$ of the $\boldsymbol{\gamma}_j$ are free to vary.

18.2.6 Stata multinomial model commands

Table 18.1 summarizes Stata commands for the estimation of multinomial models.

Table 18.1. Commands for multinomial models

Model	Command
Multinomial logit	<code>mlogit</code> , <code>xtmlogit</code>
Multinomial probit	<code>mprobit</code>
Nested logit	<code>nlogit</code>
Alternative-specific logit	<code>cmclogit</code> (formerly <code>asclogit</code>), <code>clogit</code>
Alternative-specific probit	<code>cmmprobit</code> (formerly <code>asmprobit</code>)
Alternative-specific mixed logit	<code>cmmixlogit</code> (formerly <code>asmixlogit</code>)
Ordered logit and probit	<code>ologit</code> , <code>oprobit</code> , <code>hetoprobit</code> , <code>xtologit</code> , <code>xtoprobit</code> , <code>ziologit</code> , <code>zioprobit</code>
Ordered multilevel mixed	<code>meologit</code> , <code>meoprobit</code>
Rank-ordered logit	<code>cmrologit</code> (formerly <code>rologit</code>)
Rank-ordered probit	<code>cmropprobit</code> (formerly <code>asropprobit</code>)
Stereotype logit	<code>slogit</code>
Bivariate probit	<code>biprobit</code>
Panel mixed logit	<code>cmxtmixlogit</code>

The `nlogit`, `cmclogit`, `clogit`, `cmmixlogit`, `cmmprobit`, `cmrologit`, `cmropprobit`, `cmmixlogit`, and `cmxtmixlogit` commands require the data to be in long form. The remaining commands expect data to be in wide form. The independent variable lists for most but not all of these estimation commands allow factor variables. The `margins` postestimation command is available after all but the `nlogit` and `cmrologit` commands.

18.3 Multinomial example: Choice of fishing mode

We analyze data on individual choice of whether to fish using one of four possible modes: from the beach, the pier, a private boat, or a charter boat. One explanatory variable is case specific (`income`), and the others (`price` and `crate` [catch rate]) are alternative specific.

18.3.1 Data description

The data from [Herriges and Kling \(1999\)](#) are also analyzed in [Cameron and Trivedi \(2005\)](#), chap. 15). `mus218hk.dta` has the following data:

Contains data from <code>mus218hk.dta</code>				
Observations:	1,182 <th data-cs="3" data-kind="parent">A.C.Cameron & P.K.Trivedi (2022): Microeconometrics Using Stata, 2e</th> <th data-kind="ghost"></th> <th data-kind="ghost"></th>	A.C.Cameron & P.K.Trivedi (2022): Microeconometrics Using Stata, 2e		
Variables:	16	1 Sep 2020 16:38		
Variable name	Storage type	Display format	Value label	Variable label
mode	float	%9.0g	modetype	Fishing mode
price	float	%9.0g		Price for chosen alternative
crate	float	%9.0g		Catch rate for chosen alternative
dbeach	float	%9.0g		Beach mode chosen
dpier	float	%9.0g		Pier mode chosen
dprivate	float	%9.0g		Private boat mode chosen
dcharter	float	%9.0g		Charter boat mode chosen
pbeach	float	%9.0g		Price for beach mode
ppier	float	%9.0g		Price for pier mode
pprivate	float	%9.0g		Price for private boat mode
pcharter	float	%9.0g		Price for charter boat mode
qbeach	float	%9.0g		Catch rate for beach mode
qpier	float	%9.0g		Catch rate for pier mode
qprivate	float	%9.0g		Catch rate for private boat mode
qcharter	float	%9.0g		Catch rate for charter boat mode
income	float	%9.0g		Monthly income in thousands \$

Sorted by:

There are 1,182 observations, one per individual. The first three variables are for the chosen fishing mode with the variables `mode`, `price`, and `crate` being, respectively, the chosen fishing mode and the price and catch rate for that mode. The next four variables are mutually exclusive dummy variables for the chosen mode, taking on a value of 1 if that alternative is chosen and a value of 0 otherwise. The next eight variables are alternative-specific variables that contain the price and catch rate for each of the four possible fishing modes (the prefix `q` stands for quality; a higher catch rate implies a higher quality of fishing). These variables are constructed from individual surveys that ask not only about attributes of the chosen fishing mode but also about attributes of alternative fishing modes such as location that allow for determination of price and catch rate. The final variable, `income`, is a case-specific variable. The summary statistics follow:

```
. * Summarize dependent variable and regressors
. summarize, separator(0)
```

Variable	Obs	Mean	Std. dev.	Min	Max
mode	1,182	3.005076	.9936162	1	4
price	1,182	52.08197	53.82997	1.29	666.11
crate	1,182	.3893684	.5605964	.0002	2.3101
dbeach	1,182	.1133672	.3171753	0	1
dpier	1,182	.1505922	.3578023	0	1
dprivate	1,182	.3536379	.4783008	0	1
dcharter	1,182	.3824027	.4861799	0	1
pbeach	1,182	103.422	103.641	1.29	843.186
ppier	1,182	103.422	103.641	1.29	843.186
pprivate	1,182	55.25657	62.71344	2.29	666.11
pcharter	1,182	84.37924	63.54465	27.29	691.11
qbeach	1,182	.2410113	.1907524	.0678	.5333
qpier	1,182	.1622237	.1603898	.0014	.4522
qprivate	1,182	.1712146	.2097885	.0002	.7369
qcharter	1,182	.6293679	.7061142	.0021	2.3101
income	1,182	4.099337	2.461964	.4166667	12.5

The variable `mode` takes on the values ranging from 1 to 4. On average, private and charter boat fishing are less expensive than beach and pier fishing. Beach and pier fishing, both close to shore with similar costs, have identical prices. The catch rate for charter boat fishing is substantially higher than for the other modes.

The `tabulate` command gives the various values and frequencies of the `mode` variable. We have

```
. * Tabulate the dependent variable
. tabulate mode
```

Fishing mode	Freq.	Percent	Cum.
Beach	134	11.34	11.34
Pier	178	15.06	26.40
Private	418	35.36	61.76
Charter	452	38.24	100.00
Total	1,182	100.00	

The shares are roughly one-third fish from the shore (either beach or pier), one-third fish from a private boat, and one-third fish from a charter boat.

These shares are the same as the means of `dbeach`, ..., `dcharter` given in the `summarize` table. The `mode` variable takes on a value from 1 to 4 (see the summary statistics), but the output of `describe` has a label, `modetype`, that labels 1 as `Beach`, ..., 4 as `Charter`. This labeling can be verified by using the `label list` command. There is no obvious ordering of the fishing modes, so unordered multinomial models should be used to explain fishing-mode choice.

18.3.2 Case-specific regressors

Before formal modeling, it is useful to summarize the relationship between the dependent variable and the regressors. This is more difficult when the dependent variable is an unordered dependent variable.

For the case-specific `income` variable, we could use the `bysort mode: summarize income` command. More compact output is obtained by instead using the `table` command. We obtain

```
. * Table of income by fishing mode
. table mode, stat(count income) stat(mean income) stat(sd income)
```

	Number of nonmissing values	Mean	Standard deviation
Fishing mode			
Beach	134	4.051617	2.50542
Pier	178	3.387172	2.340324
Private	418	4.654107	2.777898
Charter	452	3.8809	2.050028
Total	1,182	4.099337	2.461964

On average, those fishing from the pier have the lowest income and those fishing from a private boat have the highest.

18.3.3 Alternative-specific regressors

The relationship between the chosen fishing mode and the alternative-specific regressor price is best summarized as follows:

```
. * Table of fishing price by fishing mode
. table (result) mode, stat(mean pbeach ppier pprivate pcharter) nformat(%6.0f)
```

	Fishing mode				
	Beach	Pier	Private	Charter	Total
Price for beach mode	36	31	138	121	103
Price for pier mode	36	31	138	121	103
Price for private boat mode	98	82	42	45	55
Price for charter boat mode	125	110	71	75	84

On average, individuals tend to choose the fishing mode that is the cheapest or second-cheapest alternative available for them. For example, for those choosing `private`, on average, the price of private boat fishing is 42, compared with 71 for charter boat fishing and 138 for beach or pier fishing.

Similarly, for the catch rate, we have

```
. * Table of fishing catch rate by fishing mode
. table (result) mode, stat(mean qbeach qpier qprivate qcharter) nformat(%6.2f)
```

	Fishing mode				
	Beach	Pier	Private	Charter	Total
Catch rate for beach mode	0.28	0.26	0.21	0.25	0.24
Catch rate for pier mode	0.22	0.20	0.13	0.16	0.16
Catch rate for private boat mode	0.16	0.15	0.18	0.18	0.17
Catch rate for charter boat mode	0.52	0.50	0.65	0.69	0.63

The chosen fishing mode is not on average that with the highest catch rate. In particular, the catch rate is always highest on average for charter fishing, regardless of the chosen mode. Regression analysis can measure the effect of the catch rate after controlling for the price of the fishing mode.

18.4 Multinomial logit model

Many multinomial studies are based on datasets that have only case-specific variables because explanatory variables are typically observed only for the chosen alternative and not for the other alternatives. The simplest model is the MNL model because computation is simple and parameter estimates are easier to interpret than in some other multinomial models.

18.4.1 The mlogit command

The MNL model can be used when all the regressors are case specific. The MNL model specifies that

$$p_{ij} = \frac{\exp(\mathbf{x}'_i \boldsymbol{\beta}_j)}{\sum_{l=1}^m \exp(\mathbf{x}'_i \boldsymbol{\beta}_l)}, \quad j = 1, \dots, m \quad (18.5)$$

where \mathbf{x}_i are case-specific regressors, here an intercept and income. Clearly, this model ensures that $0 < p_{ij} < 1$ and $\sum_{j=1}^m p_{ij} = 1$. To ensure model identification, $\boldsymbol{\beta}_j$ is set to zero for one of the categories, and coefficients are then interpreted with respect to that category, called the base category.

The `mlogit` command has the syntax

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

where *indepvars* are the case-specific regressors and the default is to automatically include an intercept. The `baseoutcome(#)` option specifies the value of *depvar* to be used as the base category, overriding the Stata default of setting the most frequently chosen category as the base category. Other options include `rrr` to report exponentiated coefficients ($e^{\hat{\beta}}$ rather than $\hat{\beta}$).

The `mlogit` command requires that data be in wide form, with one observation per individual. This is the case here.

18.4.2 Application of the mlogit command

We regress fishing mode on an intercept and income, the only case-specific regressor in our dataset. There is no natural base category. The first category, beach fishing, is arbitrarily set to be the base category. We obtain

```
. * Multinomial logit with base outcome alternative 1
. mlogit mode income, baseoutcome(1) nolog vce(robust)

Multinomial logistic regression                                         Number of obs = 1,182
                                                               Wald chi2(3) = 34.13
                                                               Prob > chi2 = 0.0000
Log pseudolikelihood = -1477.1506                                         Pseudo R2 = 0.0137
```

mode	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
Beach	(base outcome)					
Pier						
	income	-.1434029	.0608337	-2.36	0.018	-.2626348
Private	_cons	.8141503	.2506076	3.25	0.001	.3229684
	income	.0919064	.0421603	2.18	0.029	.0092737
Charter	_cons	.7389208	.2017495	3.66	0.000	.343499
	income	-.0316399	.0424705	-0.74	0.456	-.1148805
	_cons	1.341291	.1971834	6.80	0.000	.9548192
						1.727764

The model fit is poor with pseudo- R^2 , defined in section 13.8.1, equal to 0.014. Three sets of regression estimates are given, corresponding here to $\hat{\beta}_2$, $\hat{\beta}_3$, and $\hat{\beta}_4$, because we used the normalization $\beta_1 = 0$.

Of the three coefficient estimates of `income`, two are statistically significant at the 0.05 level, but the results of such individual testing will vary with the omitted category. Instead, we should perform a joint test. Using a Wald test, we obtain

```

. * Wald test of the joint significance of income
. test income
( 1) [Beach]o.income = 0
( 2) [Pier]income = 0
( 3) [Private]income = 0
( 4) [Charter]income = 0
      Constraint 1 dropped
      chi2( 3) =    34.13
      Prob > chi2 =    0.0000

```

Income is clearly highly statistically significant. Note that this test is identical to the overall test `Wald chi2(3)=34.13`, which was included in the original regression output.

18.4.3 Coefficient interpretation

Coefficients in a multinomial model can be interpreted in the same way as binary logit model parameters are interpreted, with comparison being with the base category.

This is a result of the MNL model being equivalent to a series of pairwise logit models. For simplicity, we set the base category to be the first category. Then the MNL model defined in [\(18.5\)](#) implies that

$$\Pr(y_i = j | y_i = j \text{ or } 1) = \frac{\Pr(y_i = j)}{\Pr(y_i = j) + \Pr(y_i = 1)} = \frac{\exp(\mathbf{x}'_i \boldsymbol{\beta}_j)}{1 + \exp(\mathbf{x}'_i \boldsymbol{\beta}_j)}$$

using $\boldsymbol{\beta}_1 = \mathbf{0}$ and cancellation of $\sum_{l=1}^m \exp(\mathbf{x}'_i \boldsymbol{\beta}_l)$ in the numerator and denominator.

Thus, $\hat{\boldsymbol{\beta}}_j$ can be viewed as parameters of a binary logit model between alternative j and alternative 1. So a positive coefficient from `mlogit` means that as the regressor increases, we are more likely to choose alternative j than alternative 1. This interpretation will vary with the base category and is clearly most useful when there is a natural base category.

Some researchers find it helpful to transform to odds ratios or relative-risk ratios, as in the binary logit case. The odds ratio or relative-risk ratio of choosing alternative j rather than alternative 1 is given by

$$\frac{\Pr(y_i = j)}{\Pr(y_i = 1)} = \exp(\mathbf{x}'_i \boldsymbol{\beta}_j) \quad (18.6)$$

so $e^{\boldsymbol{\beta}_{jr}}$ gives the proportionate change in the relative risk of choosing alternative j rather than alternative 1 when x_{ir} changes by one unit.

The `rrr` option of `mlogit` provides coefficient estimates transformed to relative-risk ratios. We have

```
. * Relative-risk option reports exp(b) rather than b
. mlogit mode income, rrr baseoutcome(1) nolog vce(robust)

Multinomial logistic regression                               Number of obs = 1,182
                                                               Wald chi2(3) = 34.13
                                                               Prob > chi2 = 0.0000
Log pseudolikelihood = -1477.1506                         Pseudo R2 = 0.0137
```

mode	Robust					
	RRR	std. err.	z	P> z	[95% conf. interval]	
Beach	(base outcome)					
Pier						
income	.8664049	.0527066	-2.36	0.018	.7690227	.9761187
_cons	2.257257	.5656857	3.25	0.001	1.381222	3.688914
Private						
income	1.096262	.0462187	2.18	0.029	1.009317	1.190697
_cons	2.093675	.4223978	3.66	0.000	1.409872	3.109129
Charter						
income	.9688554	.0411478	-0.74	0.456	.8914726	1.052955
_cons	3.823979	.754025	6.80	0.000	2.598201	5.628054

Note: `_cons` estimates baseline relative risk for each outcome.

```
. estimates store MNL
```

Thus, a one-unit increase in `income`, corresponding to a \$1,000 monthly increase, leads to a relative odds of choosing to fish from a pier rather than

the beach that are 0.866 times what the relative odds were before the change; so the relative odds has declined. The original coefficient of `income` for the alternative `pier` was -0.1434 and $e^{-0.1434} = 0.8664$.

18.4.4 Predicted probabilities

After most estimation commands, the `predict` command creates one variable. After `mlogit`, however, m variables are created, where m is the number of alternatives. Predicted probabilities for each alternative are obtained by using the `pr` option of `predict`.

Here we obtain four predicted probabilities because there are four alternatives. We have

```
. * Predict probabilities of choice of each mode, and compare to actual freqs
. predict pmlogit1 pmlogit2 pmlogit3 pmlogit4, pr
. summarize pmlogit* dbeach dpier dprivate dcharter, separator(4)
```

Variable	Obs	Mean	Std. dev.	Min	Max
pmlogit1	1,182	.1133672	.0036716	.0947395	.1153659
pmlogit2	1,182	.1505922	.0444575	.0356142	.2342903
pmlogit3	1,182	.3536379	.0797714	.2396973	.625706
pmlogit4	1,182	.3824027	.0346281	.2439403	.4158273
dbeach	1,182	.1133672	.3171753	0	1
dpier	1,182	.1505922	.3578023	0	1
dprivate	1,182	.3536379	.4783008	0	1
dcharter	1,182	.3824027	.4861799	0	1

Note that the sample average predicted probabilities equal the observed sample frequencies. This is always the case for MNL models that include an intercept, generalizing the similar result for binary logit models.

The ideal multinomial model will predict perfectly. For example, `p1` ideally would take on a value of 1 for the 134 observations with $y = 1$ and would take on a value of 0 for the remaining observations. Here `p1` ranges only from 0.0947 to 0.1154, so the model with `income` as the only explanatory variable predicts beach fishing very poorly. There is considerably more variation in predicted probabilities for the other three alternatives.

The `margins` command (see section 13.7) can be used to compute the average predicted probability of a given outcome, along with an associated confidence interval. For example, for the third outcome, we have

```
. * Sample average predicted probability of the third outcome
. margins, predict(outcome(3)) noatlegend
Predictive margins                                         Number of obs = 1,182
Model VCE: Robust
Expression: Pr(mode==Private), predict(outcome(3))


```

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
_cons	.3536379	.0137189	25.78	0.000	.3267493	.3805265

18.4.5 MEs

For an unordered multinomial model, there is no single conditional mean of the dependent variable, y . Instead there are m alternatives, and we model the probabilities of these alternatives. Interest lies in how these probabilities change as regressors change.

For the MNL model, the MES can be shown to be

$$\frac{\partial p_{ij}}{\partial \mathbf{x}_i} = p_{ij} (\boldsymbol{\beta}_j - \bar{\boldsymbol{\beta}}_i)$$

where $\bar{\boldsymbol{\beta}}_i = \sum_l p_{il} \boldsymbol{\beta}_l$ is a probability weighted average of the $\boldsymbol{\beta}_l$. The MES vary with the point of evaluation, \mathbf{x}_i , because p_{ij} varies with \mathbf{x}_i . The signs of the regression coefficients do not give the signs of the MES. For a variable x_{ir} , the ME is positive if $\beta_{jr} > \bar{\beta}_{ir}$.

The `margins, dydx()` command calculates the average marginal effect (AME), the marginal effect at the mean (MEM), and the marginal effect at representative values (MER). By default, the MES are computed for all outcomes. In the following examples, we compute MES for a single outcome.

For example, to obtain the AME on $\Pr(y = 3)$ of a change in income, we use `predict(outcome(3))`. We obtain

```
. * AME of income change for outcome 3
. margins, dydx(*) predict(outcome(3)) noatlegend
Average marginal effects                                         Number of obs = 1,182
Model VCE: Robust
Expression: Pr(mode==Private), predict(outcome(3))
dy/dx wrt: income
```

	Delta-method				
	dy/dx	std. err.	z	P> z	[95% conf. interval]
income	.0317562	.0052485	6.05	0.000	.0214694 .0420429

Averaged across all individuals, a one-unit change in `income`, equivalent to a \$1,000 increase in monthly income, increases by 0.0318 the probability of fishing from a private boat rather than from a beach, pier, or charter boat.

To instead obtain the ME evaluated at the sample mean of regressors, we add the `atmeans` option to obtain

```
. * MEM of income change for outcome 3
. margins, dydx(*) predict(outcome(3)) atmeans noatlegend
Conditional marginal effects                                         Number of obs = 1,182
Model VCE: Robust
Expression: Pr(mode==Private), predict(outcome(3))
dy/dx wrt: income
```

	Delta-method				
	dy/dx	std. err.	z	P> z	[95% conf. interval]
income	.0325985	.00567	5.75	0.000	.0214856 .0437115

For the average individual, a one-unit change in `income` increases by 0.0326 the probability of fishing from a private boat rather than from the other fishing sites. The AME and MEM are quite similar in this example. Usually, there is greater difference in these ME measures following `mlogit` estimation.

18.5 Alternative-specific conditional logit model

Some multinomial studies use richer datasets that include alternative-specific variables, such as prices and quality measures for all alternatives, not just the chosen alternative. The CL model is used with these data.

18.5.1 Creating long-form data from wide-form data

The parameters of CL models are fit with commands that require the data to be in long form, with one observation providing the data for just one alternative for an individual.

Some datasets will already be in long form, but that is not the case here. Instead, `mus218hk.dta` is in wide form, with one observation containing data for all four alternatives for an individual. For example,

```
. * Data are in wide form
. list mode price pbeach ppier pprivate pcharter in 1, clean
      mode    price    pbeach    ppier    pprivate    pcharter
 1.   Charter    182.93    157.93    157.93    157.93    182.93
```

The first observation has data for the price of all four alternatives. The chosen mode was `charter`, so `price` was set to equal `pcharter`.

To convert data from wide form to long form, we use the `reshape` command, introduced in section 8.10. Here the long form will have four observations for each individual according to whether the suffix is `beach`, `pier`, `private`, or `charter`. These suffixes are strings, rather than the `reshape` command's default of numbers, so we use `reshape` with the `string` option. For completeness, we actually provide the four suffixes.

```

. * Convert data from wide form to long form
. generate id = _n
. reshape long d p q, i(id) j(fishmode beach pier private charter) string
Data                               Wide    ->    Long
Number of observations           1,182    ->   4,728
Number of variables                 22     ->    14
j variable (4 values)          -> fishmode
xij variables:
      dbeach dpier ... dcharter -> d
      pbeach ppier ... pcharter -> p
      qbeach qpier ... qcharter -> q

```

```

. save mus218hklong, replace
(file mus218hklong.dta not found)
file mus218hklong.dta saved

```

There are now four observations for the first individual or case. If we had not provided the four suffixes, the `reshape` command would have erroneously created a fifth alternative, `rice`, from `price` that like `pbeach`, `ppier`, `pprivate`, and `pcharter` also begins with the letter `p`.

To view the resulting long-form data for the first individual case, we list the first four observations.

```

. * List data for the first case after reshape
. list in 1/4, clean noobs
      id  fishmode    mode    price   crate    d      p      q      income
> _est_MNL  pmlogit1  pmlogit2  pmlogit3  pmlogit4
      1    beach    Charter   182.93   .5391     0   157.93   .0678   7.083332
>      1   .1125092   .0919656   .4516733   .3438518
      1    charter   Charter   182.93   .5391     1   182.93   .5391   7.083332
>      1   .1125092   .0919656   .4516733   .3438518
      1    pier     Charter   182.93   .5391     0   157.93   .0503   7.083332
>      1   .1125092   .0919656   .4516733   .3438518
      1    private   Charter   182.93   .5391     0   157.93   .2601   7.083332
>      1   .1125092   .0919656   .4516733   .3438518

```

The order is no longer pier, beach, private boat, and then charter boat. Instead, it is now beach, charter boat, pier, and then private boat because the observations are sorted in the alphabetical order of `fishmode`. For this first observation, the outcome variable, `d`, equals 1 for charter boat fishing, as expected. The four separate observations on the alternative-specific variables, `p` and `q`, are the different values for price and quality for the four alternatives.

All case-specific variables appear as a single variable that takes on the same value for the four outcomes. For `income`, this is no problem. But `mode`, `price`, and `crate` are misleading here. The `mode` variable indicates that for case 1, the fishing mode was `mode=4` because in original wide form, this corresponded to charter boat fishing. But `d=1` for the second observation of the first case because this corresponds to charter boat fishing in the reordered long form. It would be best to simply drop the misleading variables by typing `drop mode price crate` because these variables are not needed.

18.5.2 The `cmset` command

Most models with alternative-specific regressors are estimated using `cm` commands for choice models. These commands require first using the `cmset` command, which has the syntax

```
cmset caseidvar altvar [, force]
```

where `caseidvar` identifies the case (individual) and `altvar` identifies the alternatives (choice sets).

For the current example, we have

```
. * cmset before use of cm commands for alternative-specific regressors
. cmset id fishmode
      Case ID variable: id
      Alternatives variable: fishmode
```

The `cmsummarize` command provides data summary by chosen alternative. The following example provides the number of observations and the mean for the regressor variables.

```
. * Summarize the data by chosen alternative specific
. cmsummarize p q income, choice(d) statistics(N mean)
Statistics by chosen alternatives (d = 1)

income is constant within case
```

Summary statistics: N, Mean
 Group variable: _chosen_alternative (d = 1)

_chosen_alternative	p	q	income
beach	134 35.69949	134 .2791948	134 4.051617
charter	452 75.09694	452 .6914998	452 3.8809
pier	178 30.57133	178 .2025348	178 3.387172
private	418 41.60681	418 .1775411	418 4.654107
Total	1182 52.08197	1182 .3893684	1182 4.099337

The `cmtab` command provides tabulations rather than summary statistics. The `cmchoiceset` command provides sample sizes for the choice sets facing individuals and is useful when, unlike the current example, the data are unbalanced with different individuals facing different choice sets. The `cmsample` command lists the reasons why observations in a choice model were dropped from the estimation sample.

18.5.3 The `cmclogit` command

When some or all regressors are alternative specific, the CL model is used. The CL model specifies that

$$p_{ij} = \frac{\exp(\mathbf{x}'_{ij}\boldsymbol{\beta} + \mathbf{z}'_i\boldsymbol{\gamma}_j)}{\sum_{l=1}^m \exp(\mathbf{x}'_{il}\boldsymbol{\beta} + \mathbf{z}'_i\boldsymbol{\gamma}_l)}, \quad j = 1, \dots, m \quad (18.7)$$

where x_{ij} are alternative-specific regressors and z_i are case-specific regressors. To ensure model identification, we set one of the γ_j to zero, as with the MNL model. Some authors call the model above a mixed logit model, with CL used to refer to a more restrictive model that has only alternative-specific regressors.

The `cmclogit` command, an acronym for choice-model (or alternative-specific) conditional logit, has the syntax

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

where *indepvars* are the alternative-specific regressors. The identifier for each case or individual and the possible alternatives have already been specified in the `cmset` command.

The `casevars()` option specifies case-specific variables. The `basealternative()` option specifies the alternative that is to be used as the base category, which affects only the coefficients of case-specific regressors. The `altwise` option deletes only the data for an alternative, rather than the entire observation, if data are missing.

The `noconstant` option overrides the Stata default of including case-specific intercepts. Attributes of each alternative are then explained solely by alternative-specific regressors if `noconstant` is used. The case-specific intercepts provided by the default estimator are interpreted as reflecting the desirability of each alternative because of unmeasured attributes of the alternative.

The `cmclogit` command allows the choice set to vary across individuals and allows more than one alternative to be selected.

18.5.4 Application of the `cmclogit` command

We estimate the parameters of the CL model to explain fishing-mode choice given alternative-specific regressors on price and quality; the case-specific regressor, `income`; and case-specific intercepts. As for the MNL model, beach fishing is set to be the base category. We have

```

. * Conditional logit with alternative-specific and case-specific regressors
. cmclogit d p q, casevars(income) basealternative(beach) nolog vce(robust)

Conditional logit choice model
Case ID variable: id
Alternatives variable: fishmode
Log pseudolikelihood = -1215.1376

Number of obs      =     4,728
Number of cases   =      1182
Alts per case: min =        4
                           avg =      4.0
                           max =        4
Wald chi2(5)      =    178.32
Prob > chi2       =     0.0000
(Std. err. adjusted for clustering on id)

```

d		Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
fishmode	p	-.0251166	.0023261	-10.80	0.000	-.0296757 -.0205575
	q	.357782	.1173829	3.05	0.002	.1277157 .5878482
beach	(base alternative)					
charter	income	-.0332917	.0493558	-0.67	0.500	-.1300273 .0634438
	_cons	1.694366	.2206146	7.68	0.000	1.261969 2.126762
pier	income	-.1275771	.0547194	-2.33	0.020	-.2348252 -.020329
	_cons	.7779593	.2311108	3.37	0.001	.3249905 1.230928
private	income	.0894398	.047773	1.87	0.061	-.0041936 .1830732
	_cons	.5272788	.2106221	2.50	0.012	.1144671 .9400905

. estimates store CL

The first set of estimates are the coefficients $\hat{\beta}$ for the alternative-specific regressors price and quality. The next three sets of estimates are for the case-specific intercepts and regressors. The coefficients are, respectively, $\hat{\gamma}_{\text{charter}}$, $\hat{\gamma}_{\text{pier}}$, and $\hat{\gamma}_{\text{private}}$ because we used the normalization $\gamma_{\text{beach}} = 0$.

The output header does not give the pseudo- R^2 , but this can be computed by using the formula given in section 13.8.1. Here $\ln L_{\text{fit}} = -1215.1$, and estimation of an intercepts-only model yields $\ln L_0 = -1497.7$, so $\tilde{R}^2 = 1 - (-1215.1)/(-1497.7) = 0.189$, much higher than the 0.014 for the MNL model in section 18.4.2. The regressors `p`, `q`, and `income` are highly jointly statistically significant with `Wald`

`chi2(5)=178`. The `test` command can be used for individual Wald tests, or the `lrtest` command can be used for likelihood-ratio (LR) tests if estimation uses default standard errors.

The CL model in this section reduces to the MNL model in section [18.4.2](#) if $\beta_p = 0$ and $\beta_q = 0$. Using either a Wald test or an LR test, this hypothesis is strongly rejected, and the CL model is the preferred model.

18.5.5 Relationship to MNL model

The MNL and CL models are essentially equivalent. The `mlogit` command is designed for case-specific regressors and data in wide form. The `cmclogit` command is designed for alternative-specific regressors and data in long form.

The parameters of the MNL model can be fit by using `cmclogit` as the special case with no alternative-specific regressors. Thus,

```
. * MNL is CL with no alternative-specific regressors  
. cmclogit d, casevars(income) basealternative(beach) vce(robust) nolog  
(output omitted)
```

yields the same estimates as the earlier `mlogit` command. When all regressors are case specific, it is easiest to use `mlogit` with data in wide form.

Going the other way, one can estimate the parameters of a CL model using `mlogit`. This is more difficult because it requires transforming alternative-specific regressors to deviations from the base category and then imposing parameter-equality constraints. For CL models, `cmclogit` is much easier to use than `mlogit`.

18.5.6 Coefficient interpretation

Coefficients of alternative-specific regressors are easily interpreted. The alternative-specific regressor can be denoted by x_r with the coefficient β_r . The effect of a change in x_{rik} , which is the value of x_r for individual i and alternative k , is

$$\frac{\partial p_{ij}}{\partial x_{rik}} = \begin{cases} p_{ij}(1 - p_{ij})\beta_r & j = k \\ -p_{ij}p_{ik}\beta_r & j \neq k \end{cases}$$

If $\beta_r > 0$, then the own-effect is positive because $p_{ij}(1 - p_{ij})\beta_r > 0$, and the cross-effect is negative because $-p_{ij}p_{ik}\beta_r < 0$. So a positive coefficient means that if the regressor increases for one category, then that category is chosen more and other categories are chosen less, and vice versa for a negative coefficient. Here the negative price coefficient of -0.025 means that if the price of one mode of fishing increases, then demand for that mode decreases, and demand for all other modes increases, as expected. For catch rate, the positive coefficient of 0.358 means a higher catch rate for one mode of fishing increases the demand for that mode and decreases the demand for the other modes.

Coefficients of case-specific regressors are interpreted as parameters of a binary logit model against the base category; see section [18.4.3](#) for the MNL model. The income coefficients of -0.033 , -0.128 , and 0.089 mean that, relative to the probability of beach fishing, an increase in income leads to a decrease in the probability of charter boat and pier fishing and an increase in the probability of private boat fishing.

18.5.7 Predicted probabilities

Predicted probabilities can be obtained using the `predict` command with the `pr` option. This provides a predicted probability for each observation, where an observation is one alternative for one individual because the data are in long form.

To obtain predicted probabilities for each of the four alternatives, we need to summarize by `fishmode`. We use the `table` command because this gives condensed output. Much lengthier output is obtained by instead using the `bysort fishmode: summarize` command. We have

```

. * Predicted probabilities of choice of each mode and compare to actual freqs
. predict pasclogit, pr
. encode fishmode, generate(fishmode2)
. table fishmode2, stat(mean d pasclogit) stat(sd pasclogit) nototal
> nformat(%6.4f)

```

	d	Mean Pr(fishmode 1 selected)	Standard deviation Pr(fishmode 1 selected)
fishmode2			
beach	0.1134	0.1134	0.1285
charter	0.3824	0.3824	0.1566
pier	0.1506	0.1506	0.1614
private	0.3536	0.3536	0.1665

As for MNL, the sample average predicted probabilities are equal to the sample probabilities. The standard deviations of the CL model predicted probabilities (all in excess of 0.10) are much larger than those for the MNL model, so the CL model predicts better. A summary is also provided by the `estat alternatives` command.

A quite different predicted probability is that of a new alternative. This is possible for the CL model if the parameters of that model are estimated using only alternative-specific regressors, which requires use of the `noconstant` option so that case-specific intercepts are not included, and the values of these regressors are known for the new category.

For example, we may want to predict the use of a new mode of fishing that has a much higher catch rate than the currently available modes but at the same time has a considerably higher price. The parameters, β , in (18.7) are estimated with m alternatives, and then predicted probabilities are computed by using (18.7) with $m + 1$ alternatives.

18.5.8 MEs

The usual `margins` command is available after the `cmclogit` command and can be used to obtain the AME, MEM, and MER. Options for this command include the option `alternatives()`, which provides margins and MES for only the specific alternatives listed.

We compute for all alternatives the AME for just the regressor price. We obtain

```
. * AME of change in price
. margins, dydx(p)

Average marginal effects
Model VCE: Robust
Number of obs = 4,728

Expression: Pr(fishmode|1 selected), predict()
dy/dx wrt: p
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
p						
_outcome#fishmode						
beach#beach	-.0021102	.0002213	-9.54	0.000	-.0025439	-.0016765
beach #						
charter	.0006469	.0000619	10.46	0.000	.0005257	.0007682
beach#pier	.0009075	.0001373	6.61	0.000	.0006384	.0011765
beach #						
private	.0005558	.0000495	11.24	0.000	.0004588	.0006527
charter #						
beach	.0006469	.0000619	10.46	0.000	.0005257	.0007682
charter #						
charter	-.0053165	.0004529	-11.74	0.000	-.0062042	-.0044288
charter#pier	.0009157	.0000767	11.94	0.000	.0007654	.001066
charter #						
private	.0037539	.0003704	10.13	0.000	.0030279	.0044798
pier#beach	.0009075	.0001373	6.61	0.000	.0006384	.0011765
pier#charter	.0009157	.0000767	11.94	0.000	.0007654	.001066
pier#pier	-.0025593	.0002376	-10.77	0.000	-.0030249	-.0020936
pier#private	.0007361	.0000608	12.10	0.000	.0006169	.0008553
private #						
beach	.0005558	.0000495	11.24	0.000	.0004588	.0006527
private #						
charter	.0037539	.0003704	10.13	0.000	.0030279	.0044798
private#pier	.0007361	.0000608	12.10	0.000	.0006169	.0008553
private #						
private	-.0050457	.0004295	-11.75	0.000	-.0058875	-.0042039

There are 16 MES in all, corresponding to probabilities of four alternatives times prices for each of the four alternatives. All own-effects are negative and all cross-effects are positive, as explained in section [18.5.6](#). The first ME given in the output takes value -0.00211 , which means that on average a \$1 increase in the price of beach fishing decreases the probability of beach fishing by 0.00211. The second value of 0.000647 means that a \$1 increase

in the price of charter boat fishing increases beach fishing probability by 0.000647, and so on.

18.5.9 The clogit command

The CL model can also be fit by using the `clogit` command, yielding the same results. The `clogit` command is designed for grouped data used in matched case-control group studies and is similar to the `xtlogit` command used for panel data grouped over time for an individual.

The `clogit` command does not have an option for case-specific variables. Instead, a case-specific variable is interacted with dummies for $m - 1$ alternatives, and the $m - 1$ variables are entered as regressors. For applications such as the one studied in this chapter, `cmclogit` is easier to use than `clogit`.

18.5.10 Rank-ordered logit

The CL model is used when one out of m mutually exclusive alternatives is chosen. An alternative form of data is one where the m alternatives, or a subset of the m alternatives, are rank ordered.

Then one can fit an m -alternative CL model for the highest-ranked alternative, an $(m - 1)$ -alternative CL model for the second-ranked alternative, and so on. The `cmrlogit` command implements this procedure. The additional ranking data may lead to more precise estimates, though it is being assumed that the same underlying parametric model applies for the preferred alternative, the second preferred alternative, and so on.

18.6 Nested logit model

The MNL and CL models are the most commonly used multinomial models, especially in other branches of applied statistics. However, in microeconometrics applications that involve individual choice, the models are viewed as placing restrictions on individual decision making that are unrealistic, as explained below.

The simplest generalization is an NL model. Two variants of the NL model are used. The preferred variant is one based on the ARUM. This is the model we present and is the default model for Stata 17. A second variant was used by most packages in the past, including Stata before version 10. Both variants have MNL and CL as special cases, and both ensure that multinomial probabilities lie between 0 and 1 and sum to 1. But the variant based on ARUM is preferred because it is consistent with utility maximization.

18.6.1 Relaxing the independence of irrelevant alternatives assumption

The MNL and CL models impose the restriction that the choice between any two pairs of alternatives is simply a binary logit model; see (18.6). This assumption, called the independence of irrelevant alternatives (IIA) assumption, can be too restrictive, as illustrated by the “red bus/blue bus” problem. Suppose commute-mode alternatives are car, blue bus, or red bus. The IIA assumption is that the probability of commuting by car, given commuting by either car or red bus, is independent of whether commuting by blue bus is an option. But the introduction of a blue bus, same as a red bus in every aspect except color, should have little impact on car use and should halve use of red bus, leading to an increase in the conditional probability of car use given commuting by car or red bus.

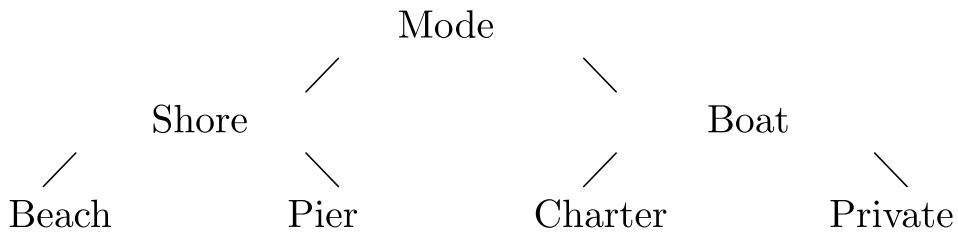
This limitation has led to alternative richer models for unordered choice based on the ARUM introduced in section 18.2.5. The MNL and CL models can be shown to arise from the ARUM if the errors, ε_{ij} , in (18.3) are independent and identically distributed as a type I extreme value. Instead, in the red bus/blue bus example, we expect the blue bus error, ε_{i2} , to be highly

correlated with the red bus error, ε_{i3} , because if we overpredict the red bus utility given the regressors, then we will also overpredict the blue bus utility.

More general multinomial models, presented in this and subsequent sections, allow for correlated errors. The NL is the most tractable of these models.

18.6.2 NL model

The NL model requires that a nesting structure be specified that splits the alternatives into groups, where errors in the ARUM are correlated within group but are uncorrelated across groups. We specify a two-level NL model, though additional levels of nesting can be accommodated, and assume a fundamental distinction between shore and boat fishing. The tree is



The shore/boat contrast is called level 1 (or a limb), and the next level is called level 2 (or a branch). The tree can be viewed as a decision tree—first decide whether to fish from shore or boat, and then decide between beach and pier (if shore) or between charter and private (if boat). But this interpretation of the tree is not necessary. The key is that the NL model permits correlation of errors within each of the level-2 groupings. Here $(\varepsilon_{i,\text{beach}}, \varepsilon_{i,\text{pier}})$ are a bivariate correlated pair, $(\varepsilon_{i,\text{private}}, \varepsilon_{i,\text{charter}})$ are a bivariate correlated pair, and the two pairs are independent. The CL model is the special case where all errors are independent.

More generally, denote alternatives by subscripts (j, k) , where j denotes the limb (level 1) and k denotes the branch (level 2) within the limb, and different limbs can have different numbers of branches, including just one branch. For example, $(2, 3)$ denotes the third alternative in the second limb. The two-level random utility is defined to be

$$U_{jk} + \varepsilon_{jk} = \mathbf{z}'_j \boldsymbol{\alpha} + \mathbf{x}'_{jk} \boldsymbol{\beta}_j + \varepsilon_{jk}, \quad j = 1, \dots, J, \quad k = 1, \dots, K_j$$

where \mathbf{z}_j varies over limbs only and \mathbf{x}_{jk} varies over both limbs and branches. For ease of exposition, we have suppressed the individual subscript i , and we consider only alternative-specific regressors. (If all regressors are instead case specific, then we have $\mathbf{z}' \boldsymbol{\alpha}_j + \mathbf{x}' \boldsymbol{\beta}_{jk} + \varepsilon_{jk}$ with one of the $\boldsymbol{\beta}_{jk} = \mathbf{0}$.) The NL model assumes that $(\varepsilon_{j1}, \dots, \varepsilon_{jK_j})$ are distributed as Gumbel's multivariate extreme-value distribution. Then the probability that alternative (j, k) is chosen equals

$$p_{jk} = p_j \times p_{k|j} = \frac{\exp(\mathbf{z}'_j \boldsymbol{\alpha} + \tau_j I_j)}{\sum_{m=1}^J \exp(\mathbf{z}'_m \boldsymbol{\alpha} + \tau_m I_m)} \times \frac{\exp(\mathbf{x}'_{jk} \boldsymbol{\beta}_j / \tau_j)}{\sum_{l=1}^{K_j} \exp(\mathbf{x}'_{jl} \boldsymbol{\beta}_j / \tau_j)}$$

where $I_j = \ln \left\{ \sum_{l=1}^{K_j} \exp(\mathbf{x}'_{jl} \boldsymbol{\beta}_j / \tau_j) \right\}$ is called the inclusive value or the log sum. The NL probabilities are the product of probabilities p_j and $p_{k|j}$, which are essentially of CL form. The model produces positive probabilities that sum to one for any value of τ_j , called dissimilarity parameters. But the ARUM restricts $0 \leq \tau_j \leq 1$, and values outside this range mean the model, while mathematically correct, is inconsistent with random-utility theory.

18.6.3 The nlogit command

The Stata commands for NL have complicated syntax that we briefly summarize. It is simplest to look at the specific application in this section and see [CM] **nlogit** for further details.

The first step is to specify the tree structure. The **nlogitgen** command has the syntax

```
nlogitgen newaltvar = altvar (branchlist) [ , [no]log ]
```

The *altvar* variable is the original variable defining the possible alternatives, and *newaltvar* is a created variable necessary for `nlogit` to know what nesting structure should be used. Here *branchlist* is

branch, *branch* [, *branch* ...]

and *branch* is

[*label:*] *alternative* [| *alternative* [| *alternative* ...]]

There must be at least two branches, and each branch has one or more alternatives.

The nesting structure can be displayed by using the `nlogittree` command with the syntax

`nlogittree` *altvarlist* [*if*] [*in*] [*weight*] [, *options*]

A useful option is `choice(depvar)`, which lists sample frequencies for each alternative.

Estimation of model parameters uses the `nlogit` command with the syntax

`nlogit` *depvar* [*indepvars*] [*if*] [*in*] [*weight*] [|| *lev1_equation* [|| *lev2_equation* ...]] ||
altvar: [*byaltvarlist*], `case(varname)` [*options*]

where *indepvars* are the alternative-specific regressors and case-specific regressors are introduced in *lev#_equation*. The syntax of *lev#_equation* is

altvar: [*byaltvarlist*] [, `base(#|lbl)` `estconst`]

`case(varname)` provides the identifier for each case (individual).

The NL commands use data in long form, as did `cmclogit`.

18.6.4 Model estimates

We first define the nesting structure by using the `nlogitgen` command. Here we define a variable, `type`, that is called `shore` for the pier and beach

alternatives and is called `boat` for the `private` and `charter` alternatives.

```
. * Define the tree for nested logit
. nlogitgen type = fishmode(shore: pier | beach, boat: private | charter)
New variable type is generated with 2 groups
label list lb_type
lb_type:
    1 shore
    2 boat
```

The tree can be checked by using the `nlogittree` command. We have

```
. * Check the tree
. nlogittree fishmode type, choice(d)

Tree structure specified for the nested logit model
type   N      fishmode   N      k
-----
shore 2364 └── beach     1182  134
            └── pier      1182  178
boat   2364 └── charter   1182  452
            └── private   1182  418
-----
                           Total  4728 1182

k = number of times alternative is chosen
N = number of observations at each level
```

The tree is as desired, so we are now ready to estimate with `nlogit`. First, list the dependent variable and the alternative-specific regressors. Then, define the level-1 equation for `type`, which here includes no regressors. Finally, define the level-2 equations that here have the regressors `income` and an intercept. We use the `notree` option, which suppresses the tree, because it was already output with the `nlogittree` command. We have

```

. * Nested logit model estimate
. nlogit d p q || type:, base(shore) || fishmode: income, case(id) notree
> vce(robust) nolog
note: the LR test for IIA will not be computed.

RUM-consistent nested logit regression          Number of obs      =      4,728
Case variable: id                            Number of cases     =      1182
Alternative variable: fishmode            Alts per case: min =         4
                                         avg =       4.0
                                         max =       4
                                         Wald chi2(5)    =     119.60
Log pseudolikelihood = -1192.4116           Prob > chi2      =     0.0000
                                                (Std. err. adjusted for clustering on id)

```

d	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
fishmode						
p	-.0267478	.0026455	-10.11	0.000	-.0319328	-.0215628
q	1.346875	.2661627	5.06	0.000	.8252062	1.868545

fishmode equations

beach						
income	0	(base)				
_cons	0	(base)				
charter						
income	-5.483238	12.83935	-0.43	0.669	-30.6479	19.68143
_cons	48.44557	101.3858	0.48	0.633	-150.267	247.1581
pier						
income	-6.441663	13.31746	-0.48	0.629	-32.54341	19.66009
_cons	39.98192	84.51765	0.47	0.636	-125.6696	205.6335
private						
income	-1.29216	2.208674	-0.59	0.559	-5.621081	3.036761
_cons	28.30816	60.99204	0.46	0.643	-91.23404	147.8504

dissimilarity parameters

/type						
shore_tau	56.08891	124.0073			-186.9609	299.1387
boat_tau	32.69379	88.47634			-140.7166	206.1042

```
. estimates store NL
```

The coefficient of variable *p* is little changed compared with the CL model, but the other coefficients changed considerably.

The NL model reduces to the CL model if the two dissimilarity parameters are both equal to 1. The dissimilarity parameters are much greater than 1, albeit not statistically different from 1. This is not an unusual finding for NL models; it means that while the model is mathematically correct, with probabilities between 0 and 1 that add up to 1, the fitted model is not consistent with the ARUM. A joint Wald test that the two dissimilarity parameters both equal 1 has $p = 0.646$, so the CL model is not rejected. At the same time, if default standard errors are used, the output includes an LR test with $p = 0.000$ that strongly rejects the CL model.

18.6.5 Predicted probabilities

The `predict` command with the `pr` option provides predicted probabilities for level 1, level 2, and so on. Here there are two levels. The first-level probabilities are for `shore` or `boat`. The second-level probabilities are for each of the four alternatives. We have

```
. * Predict level 1 and level 2 probabilities from NL model
. predict plevel1 plevel2, pr
. tabulate fishmode, summarize(plevel2)
```

fishmode	Summary of Pr(fishmode alternatives)		
	Mean	Std. dev.	Freq.
beach	.11320342	.13343115	1,182
charter	.38065864	.15748834	1,182
pier	.15075492	.16988406	1,182
private	.35538302	.1642526	1,182
Total	.25	.19693631	4,728

The average predicted probabilities for NL no longer equal the sample probabilities, but they are quite close. The variation in the predicted probabilities, as measured by the standard deviation, is essentially the same as that for the CL model predictions, given in section [18.5.7](#).

18.6.6 MEs

The `nlogit` command is one of very few commands for which the `margins` postestimation command, or the older `estat mfx` command, is not available. Instead, we compute the AMEs of a price change manually. We obtain

```
. * AME of beach price change computed manually
. preserve
. qui summarize p
. generate delta = r(sd)/1000
. qui replace p = p + delta if fishmode == "beach"
. predict pnew1 pnew2, pr
. generate dpdbeach = (pnew2 - plevel2)/delta

. tabulate fishmode, summarize(dpdbeach)
      Summary of dpdbeach
fishmode |   Mean   Std. dev.    Freq.
          |
  beach | -.00054265   .00048508    1,182
charter |  .00063417   .00054757    1,182
    pier | -.00064882   .00057087    1,182
 private |  .00055729   .00051243    1,182
          |
  Total | -1.047e-09   .00079865    4,728
. restore
```

Compared with the CL model, there is little change in the ME of beach price change on the probability of charter and private boat fishing. But now, the probability of pier fishing falls in addition to the probability of beach fishing.

18.6.7 Comparison of various MNL models

The following table summarizes key output from fitting the preceding MNL, CL, and NL models. We have

```
. * Summary statistics for the logit models
. estimates table MNL CL NL, keep(p q) stats(N ll aic bic) equation(1)
>     b(%7.3f) stfmt(%7.0f)
```

Variable	MNL	CL	NL
p		-0.025	-0.027
q		0.358	1.347
N	1182	4728	4728
ll	-1477	-1215	-1192
aic	2966	2446	2405
bic	2997	2498	2469

The information criteria, the Akaike information criterion and Bayesian information criterion, are presented in section 13.8.2; lower values are preferred. MNL is least preferred, and NL is most preferred.

In this example, the three multinomial models are actually nested, so we can choose between them by using LR tests. From the discussion of the CL and NL models, NL is again preferred to CL, which in turn is preferred to MNL.

All three models use the same amount of data. The CL and NL model entries have an N that is four times that for MNL because they use data in long form, leading to four “observations” per individual.

18.7 Multinomial probit model

The MNP model, like the NL model, allows relaxation of the IIA assumption. It has the advantage of allowing a much more flexible pattern of error correlation and does not require the specification of a nesting structure.

18.7.1 MNP

The MNP is obtained from the ARUM of section [18.2.5](#) by assuming normally distributed errors.

For the ARUM, the utility of alternative j is

$$U_{ij} = \mathbf{x}'_{ij}\boldsymbol{\beta} + \mathbf{z}'_i\boldsymbol{\gamma}_j + \varepsilon_{ij}$$

where the errors are assumed to be normally distributed, with $\varepsilon \sim N(\mathbf{0}, \Sigma)$, where $\varepsilon = (\varepsilon_{i1}, \dots, \varepsilon_{im})$.

Then, from [\(18.4\)](#), the probability that alternative j is chosen equals

$$p_{ij} = \Pr(y_i = j) = \Pr\{\varepsilon_{ik} - \varepsilon_{ij} \leq (\mathbf{x}_{ij} - \mathbf{x}_{ik})'\boldsymbol{\beta} + \mathbf{z}'_i(\boldsymbol{\gamma}_j - \boldsymbol{\gamma}_k)\}, \quad \text{for all } k \quad (18.8)$$

This is an $(m - 1)$ -dimensional integral for which there is no closed-form solution and computation is difficult. This problem did not arise for the preceding logit models because for those models, the distribution of ε is such that [\(18.8\)](#) has a closed-form solution.

When there are few alternatives, say, three or four, or when $\Sigma = \sigma^2 \mathbf{I}$, quadrature methods can be used to numerically compute the integral. Otherwise, maximum simulated likelihood (MSL), discussed below, is used.

Regardless of the method used, not all $(m + 1)m/2$ distinct entries in the error variance matrix, Σ , are identified. From [\(18.8\)](#), the model is defined for $m - 1$ error differences $(\varepsilon_{ik} - \varepsilon_{ij})$ with an $(m - 1) \times (m - 1)$ variance matrix that has $m(m - 1)/2$ unique terms. Because a variance term also needs to be normalized, there are only $\{m(m - 1)/2\} - 1$ unique terms in Σ . In practice, further

restrictions are often placed on Σ because otherwise Σ is imprecisely estimated, which can lead to imprecise estimation of β and γ .

18.7.2 The mprobit command

The `mprobit` command is the analogue of `mlogit`. It applies to models with only case-specific regressors and assumes that the alternative errors are independent standard normal so that $\Sigma = \mathbf{I}$. Here the $(m - 1)$ -dimensional integral in (18.8) can be shown to reduce to a one-dimensional integral that can be approximated by using quadrature methods.

There is little reason to use the `mprobit` command because the model is qualitatively similar to MNL; `mprobit` assumes that alternative-specific errors in the ARUM are uncorrelated, but it is much more computationally burdensome. The syntax for `mprobit` is similar to that for `mlogit`. For a regression with the alternative-invariant regressor `income`, the command is

```
. * MNP with independent errors and alternative-invariant regressors
. mprobit mode income, baseoutcome(1)
  (output omitted)
```

The output is qualitatively similar to that from `mlogit`, though parameter estimates are scaled differently, as in the binary model case. The fitted log likelihood is $-1,477.8$, very close to the $-1,477.2$ for MNL (see section 18.4.2).

18.7.3 Maximum simulated likelihood

The multinomial log likelihood is given in (18.2), where $p_{ij} = F_j(\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\theta})$ and the parameters $\boldsymbol{\theta}$ are $\beta, \gamma_1, \dots, \gamma_m$ (with one γ normalized to zero) and any unspecified entries in Σ .

Because there is no closed-form solution for $F_j(\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\theta})$ in (18.8), the log likelihood is approximated by a simulator, $\tilde{F}_j(\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\theta})$, that is based on S draws. A simple example is a frequency simulator that, given the current estimate $\hat{\boldsymbol{\theta}}$, takes S draws of $\varepsilon_i \sim N(\mathbf{0}, \hat{\Sigma})$ and lets $\tilde{F}_j(\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\theta})$ be the proportion of the S draws for which $\varepsilon_{ik} - \varepsilon_{ij} \leq (\mathbf{x}_{ij} - \mathbf{x}_{ik})'\hat{\beta} + \mathbf{z}'_i(\hat{\gamma}_j - \hat{\gamma}_k)$ for all k . This simulator is inadequate, however, because it is very noisy for low-probability events, and for the MNP model, the frequency simulator is nonsmooth in β and $\gamma_1, \dots, \gamma_m$ so that very small changes in these parameters may lead to no change in $\tilde{F}_j(\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\theta})$. Instead, the Geweke–Hajivassiliou–Keane simulator—described, for example, in Train (2009)—is used.

The MSL estimator maximizes

$$\ln L(\boldsymbol{\theta}) = \sum_{i=1}^N \sum_{j=1}^m y_{ij} \ln \tilde{F}_j(\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\theta}) \quad (18.9)$$

The usual ML asymptotic theory applies, provided that both $S \rightarrow \infty$ and $N \rightarrow \infty$, and $\sqrt{N}/S \rightarrow 0$ so that the number of simulations increases at a rate faster than \sqrt{N} . Even though default standard errors are fine for a correctly specified multinomial model, robust standard errors are numerically better when MSL is used.

The MSL estimator can, in principle, be applied to any estimation problem that entails an unknown integral. Some general results are the following: Smooth simulators should be used. Even then, some simulators are much better than others, but this is model specific. When random draws are used, they should be based on the same underlying uniform seed at each iteration because otherwise the gradient method may fail to converge simply because of different random draws (called chatter). The number of simulations may be greatly reduced for a given level of accuracy by using antithetic draws, rather than independent draws, and by using quasi-random-number sequences such as Halton sequences rather than uniform pseudorandom draws to generate uniform numbers. The benefits of using Halton and Hammersley rather than uniform draws is exposited in [Drukker and Gates \(2006\)](#). And to reduce the computational burden of gradient methods, it is best to at least use analytical first derivatives. For more explanation, see, for example, [Train \(2009\)](#) or [Cameron and Trivedi \(2005, chap. 15\)](#). The `cmmprobit` command incorporates all of these considerations to obtain the MSL estimator for the MNP model.

18.7.4 The `cmmprobit` command

The `cmmprobit` command requires data to be in long form, like the `cmlclogit` command, and it has similar syntax:

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

Estimation takes a long time because estimation is by MSL.

Several of the command's options are used to specify the error variance matrix Σ . As already noted, at most $\{m(m - 1)/2\} - 1$ unique terms in Σ are identified. The default identification method is to drop the row and column of Σ corresponding

to the first alternative (except that Σ_{11} is normalized to 1) and to set $\Sigma_{22} = 1$. These defaults can be changed by using the `basealternative()` and `scalealternative()` options. The `correlation()` and `stddev()` options are used to place further structure on the remaining off-diagonal and diagonal entries of Σ . The `correlation(unstructured)` option places no structure, the `correlation(exchangeable)` option imposes equicorrelation, the `correlation(independent)` option sets $\Sigma_{jk} = 0$ for all $j \neq k$, and the `correlation(pattern)` and `correlation(fixed)` options allow manual specification of the structure. The `stddev(homoskedastic)` option imposes $\Sigma_{jj} = 1$, the `stddev(heteroskedastic)` option allows $\Sigma_{jj} \neq 1$, and the `stddev(pattern)` and `stddev(fixed)` options allow manual specification of any structure.

Other options allow variations in MSL computations. `intmethod()` specifies whether the uniform numbers are from a Hammersley sequence (`intmethod(hammersley)`), the default, from a Halton sequence (`intmethod(halton)`), or from uniform pseudorandom draws (`intmethod(random)`). The `intpoints(S)` option sets the number of integration points to S . By default, $S = 500 + 2.5\sqrt{N\{\ln(k+5) + v\}}$, where k is the number of coefficients and v is the number of covariance parameters. S is set to twice this value in the case of `intmethod(random)`. In this latter case, one should additionally use the `intseed()` option, which sets the random-number generator seed. The `antithetics` option specifies antithetic draws to be used. This improves the accuracy. The `favor(speed|space)` option leads to favoring speed (the default) or space when generating the integration points. Other integration options are `intburn()`, `intseed()`, `nopivot`, and `initbhhh()`.

Consistent MSL estimation requires that the number of evaluation points go to infinity. Final published work should set `intpoints(#)` to a value much higher than the default value.

18.7.5 Application of the `cmmprobit` command

For simplicity, we restricted attention to a choice between three alternatives: fishing from a pier, private boat, or charter boat. The most general model with unstructured correlation and heteroskedastic errors is used. We use the `structural` option because then the variance parameter estimates are reported for the $m \times m$ error variance matrix Σ rather than the $(m - 1) \times (m - 1)$ variance matrix of the difference in errors. We have

```

. * Multinomial probit with unstructured errors when charter is dropped
. qui use mus218hkl, clear
. cmset id fishmode
    Case ID variable: id
Alternatives variable: fishmode
. drop if fishmode == "charter" | mode == 4
(2,538 observations deleted)

. cmmprobit d p q, casevars(income) correlation(unstructured) structural
> vce(robust) nolog
note: variable p has 106 cases that are not alternative-specific; there is no
within-case variability.

Multinomial probit choice model
Case ID variable: id
Alternatives variable: fishmode
Number of obs      =      2,190
Number of cases    =       730
Alts per case: min =        3
                           avg =     3.0
                           max =     3
Integration sequence:          Hammersley
Integration points:           642      Wald chi2(4)      =     13.06
Log simulated-pseudolikelihood = -482.29432      Prob > chi2      =     0.0110
                                         (Std. err. adjusted for clustering on id)

```

d	Robust						
	Coefficient	std. err.	z	P> z	[95% conf. interval]		
fishmode							
	p	-.0233365	.011336	-2.06	0.040	-.0455547	-.0011184
	q	1.398374	.5362936	2.61	0.009	.3472577	2.44949
beach	(base alternative)						
pier							
	income	-.0979907	.0413102	-2.37	0.018	-.1789572	-.0170242
	_cons	.7547783	.201365	3.75	0.000	.3601101	1.149447
private							
	income	.0412734	.0735513	0.56	0.575	-.1028846	.1854314
	_cons	.6605541	.2757544	2.40	0.017	.1200854	1.201023
/lnsigma3		.403742	.4964333	0.81	0.416	-.5692495	1.376733
/atanhr3_2		.1757771	.2335785	0.75	0.452	-.2820283	.6335826
sigma1		1	(base alternative)				
sigma2		1	(scale alternative)				
sigma3		1.497418	.743368			.56595	3.961939
rho3_2		.1739889	.2265076			-.2747813	.5605142

(fishmode=beach is the alternative normalizing location)
(fishmode=pier is the alternative normalizing scale)

As expected, utility is decreasing in price and increasing in quality (catch rate).

The base mode was automatically set to the first alternative, `beach`, so that the first row and column of Σ are set to 0, except $\Sigma_{11} = 1$. One additional variance restriction is needed, and here that is on the error variance of the second alternative, `pier`, with $\Sigma_{22} = 1$ (the alternative normalizing scale). With $m = 3$, there are $(3 \times 2)/2 - 1 = 2$ free entries in Σ : one error variance parameter, Σ_{33} , and one correlation, $\rho_{32} = \text{Cor}(\varepsilon_{i3}, \varepsilon_{i2})$. The `sigma3` output is $\sqrt{\Sigma_{33}}$, and the `rho3_2` output is ρ_{32} .

The `estat covariance` and `estat correlation` commands list the complete estimated variance matrix, $\widehat{\Sigma}$, and the associated correlation matrix. We have

```
. * Show correlations and covariance
. estat correlation
```

	beach	pier	private
beach	1.0000		
pier	0.0000	1.0000	
private	0.0000	0.1740	1.0000

```
. estat covariance
```

	beach	pier	private
beach	1		
pier	0	1	
private	0	.260534	2.242259

If the parameters of the model are instead estimated without the `structural` option, the same parameter estimates are obtained, aside from estimation error, but the covariances and correlation are given for the variance matrix of the bivariate distribution of $\varepsilon_{i2} - \varepsilon_{i1}$ and $\varepsilon_{i3} - \varepsilon_{i1}$.

18.7.6 Predicted probabilities and MEs

The `predict` postestimation command with the default `pr` option predicts p_{ij} , and predictive margins and MEs are obtained using the `margins` command. The commands are similar to those after `cmclogit`; see sections [18.5.7](#) and [18.5.8](#).

18.7.7 Rank-ordered probit

The rank-ordered probit model fits the MNP model when alternatives are rank ordered, rather than the more common case where only the most preferred

alternative is known.

Then one can fit an m -alternative MNP model for the highest-ranked alternative, an $(m - 1)$ -alternative MNP model for the second-ranked alternative, and so on. The `cmrrobit` command implements this procedure. The additional ranking data may lead to more precise estimates, though it is being assumed that the same underlying parametric model applies for the preferred alternative, the second preferred alternative, and so on.

18.8 Alternative-specific random-parameters logit

The alternative-specific random-parameters logit (RPL) model, or mixed logit model, relaxes the IIA assumption by allowing the coefficients of alternative-specific regressors in the CL model to be normally distributed or lognormally distributed. Here we estimate the parameters of the models by using individual-level data.

Quite different estimation procedures are used if the data are grouped, such as market share data; see section [18.8.4](#).

18.8.1 Alternative-specific RPL

The alternative-specific RPL model, or mixed logit model, is obtained from the ARUM of section [18.2.5](#) by assuming that the errors ε_{ij} are type II extreme-value distributed, like for the CL model, and the parameters β are normally distributed. Then the utility of alternative j is

$$\begin{aligned} U_{ij} &= \mathbf{x}'_{ij}\beta_i + \mathbf{z}'_i\gamma_j + \varepsilon_{ij} \\ &= \mathbf{x}'_{ij}\beta + \mathbf{z}'_i\gamma_j + \mathbf{x}'_{ij}\mathbf{v}_i + \varepsilon_{ij} \end{aligned}$$

where $\beta_i = \beta + \mathbf{v}_i$ and $\mathbf{v}_i \sim N(\mathbf{0}, \Sigma_\beta)$. The combined error $(\mathbf{x}'_{ij}\mathbf{v}_i + \varepsilon_{ij})$ is now correlated across alternatives, whereas the errors ε_{ij} alone were not.

Then, conditional on the unobservables \mathbf{v}_i , we have a CL model with

$$p_{ij}|\mathbf{v}_i = \frac{\exp(\mathbf{x}'_{ij}\beta + \mathbf{z}'_i\gamma_j + \mathbf{x}'_{ij}\mathbf{v}_i)}{\sum_{l=1}^m \exp(\mathbf{x}'_{il}\beta + \mathbf{z}'_i\gamma_l + \mathbf{x}'_{il}\mathbf{v}_i)}, \quad j = 1, \dots, m$$

The MLE is based on p_{ij} , which also requires integrating out \mathbf{v}_i , a high-dimensional integral.

The MSL estimator instead maximizes (18.9), where $\tilde{F}_j(\mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\theta})$ is a simulator for p_{ij} . Here the frequency simulator that makes many draws of \mathbf{v}_i from the normal given current estimates of Σ_β is a smooth simulator.

18.8.2 The cmmixlogit command

The `cmmixlogit` command computes the MSL estimator. This supplants the community-contributed `mixlogit` command (Hole 2007) presented in the first edition of this book. The data first need to be `cmset` to provide identifiers for each case (individual) and each alternative. The syntax of the `cmmixlogit` command is

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

which is similar to that for `cmmprobit`. Alternative-specific regressors with nonrandom coefficients are listed as `indepvars`. Key options are `casevars()` for case-specific variables and `random()` to identify alternative-specific variables with random coefficients. The default is for the random coefficients to be independent normally distributed. Other options allow for random coefficients that are correlated normal, lognormal, truncated normal, uniform, or triangular distributed.

The options for the MSL computations are similar to those for the `cmmprobit` command detailed in section 18.7.4. Consistent MSL estimation requires that the number of evaluation points go to infinity. Final published work should set `intpoints()` to a value much higher than the default value.

18.8.3 Application of the cmmixlogit command

We fit the same three-choice model as that used in section 18.7.5 for the MNP model, with charter fishing dropped.

The parameters for `p` are specified to be random, using the `random()` option. All other parameters are specified to be fixed; the alternative-specific variable `q` appears in the initial variable list, while the case-specific variable `income` appears in the `casevars()` option. We have

```

. * Alternative-specific mixed logit or random parameters logit estimation
. qui use mus218hklong, clear
. drop if fishmode == "charter" | mode == 4
(2,538 observations deleted)
. cmset id fishmode // caseidvar casevars
    Case ID variable: id
Alternatives variable: fishmode
. cmmixlogit d q, casevars(income) random(p) basealternative(pier)
>      vce(robust) nolog
Mixed logit choice model
Case ID variable: id
Alternatives variable: fishmode
Integration sequence:          Hammersley
Integration points:           613      Wald chi2(4) =     28.40
Log simulated-pseudolikelihood = -433.92078      Prob > chi2 =     0.0000
                                         (Std. err. adjusted for clustering on id)

```

d	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
fishmode						
q	.8633073	.8872554	0.97	0.331	-.8756813	2.602296
p	-.107416	.0287078	-3.74	0.000	-.1636823	-.0511497
/Normal						
sd(p)	.0595192	.0187898			.0320582	.1105035
beach						
income	.1203331	.0519823	2.31	0.021	.0184497	.2222165
_cons	-.7802862	.2304865	-3.39	0.001	-1.232031	-.328541
pier	(base alternative)					
private						
income	.1733836	.0773131	2.24	0.025	.0218526	.3249146
_cons	-.2199922	.318053	-0.69	0.489	-.8433647	.4033802

There is considerable variation across individuals in the effect of price. The random coefficients have a mean of -0.1074 and a standard deviation of 0.0595 , both statistically significant at the 0.05 level.

The preceding command used robust standard errors. If instead default standard errors are used, the output includes an LR test against a CL model that restricts the variance of β_{pi} to equal zero and strongly rejects this

simpler model. From output not included, the simpler CL model had a log likelihood of -466.8 compared with -433.9 for the RPL model. This leads to an LR test statistic of $2 \times [-433.9 - (-466.8)] = 65.8$.

MES can be obtained using the `margins` command. For example, with respect to price changes, we find

```
. * AMEs with respect to price
. margins, dydx(p)

Average marginal effects
Model VCE: Robust
Expression: Pr(fishmode), predict()
dy/dx wrt: p
```

Number of obs = 2,190

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
p						
_outcome#fishmode						
beach#beach	-.0122611	.0032902	-3.73	0.000	-.0187099	-.0058123
beach#pier	.0097067	.0028752	3.38	0.001	.0040714	.0153421
beach # private	.0025544	.0004443	5.75	0.000	.0016835	.0034252
pier#beach	.0097067	.0028752	3.38	0.001	.0040714	.0153421
pier#pier	-.0131526	.0033724	-3.90	0.000	-.0197624	-.0065428
pier#private	.0034458	.0005343	6.45	0.000	.0023986	.0044931
private # beach	.0025544	.0004443	5.75	0.000	.0016835	.0034252
private#pier	.0034458	.0005343	6.45	0.000	.0023986	.0044931
private # private	-.0060002	.0009187	-6.53	0.000	-.0078007	-.0041996

18.8.4 Berry–Levinson–Pakes market demand model

An extension of the random parameters logit model to market share data is the BLP model, named after its creators ([Berry, Levinsohn, and Pakes 1995](#)).

This model not only allows random parameters but also allows prices to be endogenous. The method is computationally burdensome, requiring Monte Carlo integration and a contraction mapping to yield a generalized method of moments objective function. Furthermore, the method can yield

multiple optima, and it can be difficult to find the optimum that minimizes the objective function; see [Knittel and Metaxoglou \(2014\)](#).

The community-contributed `blp` command provides estimates of the BLP model; see [Vincent \(2015\)](#) for further details.

18.9 Ordered outcome models

In some cases, categorical data are naturally ordered. An example is health status that is self-assessed as poor, fair, good, or excellent. The two standard models for such data are the ordered logit and ordered probit models.

18.9.1 Data summary

We use data from the Rand Health Insurance Experiment, described in detail in section [22.3](#). We use one year of this panel, so the data are cross-sectional data.

The ordered outcome we consider is health status that is poor or fair ($y = 1$), good ($y = 2$), or excellent ($y = 3$). This variable needs to be constructed from several binary outcomes for each of the health statuses. The categories poor and fair are combined because only 1.5% of the sample reports poor health. The data are constructed as follows:

```
. * Create multinomial ordered outcome variables that take values y = 1, 2, 3
. qui use mus218rchie, clear
. qui keep if year == 2
. generate hlthpf = hlthp + hlthf
. generate hlthe = (1 - hlthpf - hlthg)
. qui generate hlthstat = 1 if hlthpf == 1
. qui replace hlthstat = 2 if hlthg == 1
. qui replace hlthstat = 3 if hlthe == 1
. label variable hlthstat "Health status"
. label define hsvalue 1 "Poor or fair" 2 "Good" 3 "Excellent"
. label values hlthstat hsvalue
. tabulate hlthstat
```

Health status	Freq.	Percent	Cum.
Poor or fair	523	9.38	9.38
Good	2,034	36.49	45.87
Excellent	3,017	54.13	100.00
Total	5,574	100.00	

Health status is poor or fair for roughly 10% of the sample, good for 35%, and excellent for 55%.

The regressors considered are age in years (`age`), log annual family income (`linc`), and number of chronic diseases (`ndisease`). Summary statistics are

```
. * Summarize dependent and explanatory variables
. summarize hlthstat age linc ndisease
```

Variable	Obs	Mean	Std. dev.	Min	Max
hlthstat	5,574	2.447435	.659524	1	3
age	5,574	25.57613	16.73011	.0253251	63.27515
linc	5,574	8.696929	1.220592	0	10.28324
ndisease	5,574	11.20526	6.788959	0	58.6

The sample is of children and adults but not the elderly.

18.9.2 Ordered outcomes

The ordered outcomes are modeled to arise sequentially as a latent variable, y^* , crosses progressively higher thresholds. In the current example, y^* is an unobserved measure of healthiness. For individual i , we specify

$$y_i^* = \mathbf{x}'_i \boldsymbol{\beta} + u_i \quad (18.10)$$

where a normalization is that the regressors \mathbf{x} do not include an intercept. For very low y^* , health status is poor or fair; for $y^* > \alpha_1$, health status improves to good; for $y^* > \alpha_2$, it improves further to excellent; and so on, if there were additional categories.

For an m -alternative ordered model, we define

$$y_i = j \quad \text{if } \alpha_{j-1} < y_i^* \leq \alpha_j, \quad j = 1, \dots, m$$

where $\alpha_0 = -\infty$ and $\alpha_m = \infty$. Then,

$$\begin{aligned}
\Pr(y_i = j) &= \Pr(\alpha_{j-1} < y_i^* \leq \alpha_j) \\
&= \Pr(\alpha_{j-1} < \mathbf{x}'_i \boldsymbol{\beta} + u_i \leq \alpha_j) \\
&= \Pr(\alpha_{j-1} - \mathbf{x}'_i \boldsymbol{\beta} < u_i \leq \alpha_j - \mathbf{x}'_i \boldsymbol{\beta}) \\
&= F(\alpha_j - \mathbf{x}'_i \boldsymbol{\beta}) - F(\alpha_{j-1} - \mathbf{x}'_i \boldsymbol{\beta})
\end{aligned}$$

where F is the cumulative distribution function of u_i . The regression parameters, $\boldsymbol{\beta}$, and the $m - 1$ threshold parameters, $\alpha_1, \dots, \alpha_{m-1}$, are obtained by maximizing the log likelihood with $p_{ij} = \Pr(y_i = j)$ as defined above. Stata excludes an intercept from the regressors. If instead an intercept is estimated, then only $m - 2$ threshold parameters are identified.

For the ordered logit model, u is logistically distributed with $F(z) = e^z / (1 + e^z)$. For the ordered probit model, u is standard normally distributed with $F(\cdot) = \Phi(\cdot)$, the standard normal cumulative distribution function.

The sign of the regression parameters, $\boldsymbol{\beta}$, can be immediately interpreted as determining whether the latent variable, y^* , increases with the regressor. If β_j is positive, then an increase in x_{ij} necessarily decreases the probability of being in the lowest category ($y_i = 1$) and increases the probability of being in the highest category ($y_i = m$).

18.9.3 Application of the ologit command

The parameters of the ordered logit model are estimated by using the `ologit` command, which has syntax essentially the same as `mlogit`:

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

Application of this command yields

```

. * Ordered logit estimates
. ologit hlthstat age linc ndisease, nolog vce(robust)
Ordered logistic regression
Number of obs = 5,574
Wald chi2(3) = 665.16
Prob > chi2 = 0.0000
Pseudo R2 = 0.0720
Log pseudolikelihood = -4769.8525

```

hlthstat	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
age	-.0292944	.0016646	-17.60	0.000	-.0325568	-.0260319
linc	.2836537	.0280031	10.13	0.000	.2287687	.3385387
ndisease	-.0549905	.0041023	-13.40	0.000	-.0630308	-.0469503
/cut1	-1.39598	.2505042			-1.886959	-.9050004
/cut2	.9513097	.2500833			.4611555	1.441464

The latent health-status variable is increasing in income and decreasing with age and number of chronic diseases, as expected. The regressors are highly statistically significant. The threshold parameters appear to be statistically significantly different from each other, so the three categories should not be collapsed into two categories.

18.9.4 Predicted probabilities

Predicted probabilities for each of the three outcomes can be obtained by using the `pr` option. For comparison, we also compute the sample frequencies of each outcome.

```

. * Calculate predicted probability that y=1, 2, or 3 for each person
. predict p1ologit p2ologit p3ologit, pr
. summarize hlthpf hlthg hlthe p1ologit p2ologit p3ologit, separator(0)

```

Variable	Obs	Mean	Std. dev.	Min	Max
hlthpf	5,574	.0938285	.2916161	0	1
hlthg	5,574	.3649085	.4814477	0	1
hlthe	5,574	.541263	.4983392	0	1
p1ologit	5,574	.0946903	.0843148	.0233629	.859022
p2ologit	5,574	.3651672	.0946158	.1255265	.5276064
p3ologit	5,574	.5401425	.1640575	.0154515	.7999009

The average predicted probabilities are within 0.01 of the sample frequencies for each outcome.

18.9.5 MEs

The ME on the probability of choosing alternative j when regressor x_r changes is given by

$$\frac{\partial \Pr(y_i = j)}{\partial x_{ri}} = \{F'(\alpha_{j-1} - \mathbf{x}'_i \boldsymbol{\beta}) - F'(\alpha_j - \mathbf{x}'_i \boldsymbol{\beta})\}\beta_r$$

If one coefficient is twice as big as another, then so too is the size of the ME.

We use the `margins` command with the `atmeans` option to obtain the MEM for the third outcome (health status excellent). We obtain

```
. * MEM for third outcome (health status excellent)
. margins, dydx(*) predict(outcome(3)) atmeans noatlegend
Conditional marginal effects                                         Number of obs = 5,574
Model VCE: Robust
Expression: Pr(hlthstat==3), predict(outcome(3))
dy/dx wrt: age linc ndisease
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
age	-.0072824	.0004139	-17.59	0.000	-.0080937	-.0064712
linc	.070515	.0069821	10.10	0.000	.0568304	.0841997
ndisease	-.0136704	.0010203	-13.40	0.000	-.0156701	-.0116707

For the average individual, the probability of excellent health decreases with age or increase in diseases and increases as income increases.

The AME can be computed by dropping the `atmeans` option.

18.9.6 Other ordered models

The parameters of the ordered probit model are estimated by using the `oprobit` command. The command syntax and output are essentially the same as for ordered logit, except that coefficient estimates are scaled differently.

Application to the data here yields t statistics and log likelihoods quite close to those from ordered logit.

The `hetoprobit` command fits an ordered probit model where the error term u_i in (18.10) can be heteroskedastic. For example,

```
. * MNP with independent errors and alternative-invariant regressors  
. hetoprobit hlthstat age linc ndisease, het(age linc ndisease)  
(output omitted)
```

Compared with ordered probit, there was a substantial increase in the log likelihood from $-4,771.04$ to $-4,740.65$, leading to strong support for the heteroskedastic probit model.

The `zioprobit` command fits a zero-inflated version of the ordered probit model. It is used when a disproportionate fraction of the ordered outcomes fall into the lowest category. The lowest category outcomes are modeled as arising from both a probit model and an ordered probit model, with regressors that may differ in the two models. The `zilogit` command estimates a similar model for the ordered logit model. The community-contributed `gologit2` command ([Williams 2006](#)) fits a generalization of the ordered logit model that allows the threshold parameters $\alpha_1, \dots, \alpha_{m-1}$ to depend on regressors.

An alternative model is the MNL model. Although the MNL model has more parameters, the ordered logit model is not nested within the MNL. Estimator efficiency is another way of comparing the two approaches. An ordered estimator makes more assumptions than an MNL estimator. If these additional assumptions are true, the ordered estimator is more efficient than the MNL estimator.

18.10 Clustered data

Section 13.9 presented in some detail various models and methods that can be applied when observations in the same cluster are correlated while observations in different clusters are uncorrelated. That discussion was illustrated using the Poisson model. Here we provide a brief summary of adaptation to multinomial models.

Unless a multinomial model explicitly accounts for within cluster correlation, the estimates of a multinomial model will be inconsistent given clustering because the parametric model for $F_j(\mathbf{x}_i, \boldsymbol{\theta})$ will be misspecified. For the simplest multinomial model with clustered data, one might nonetheless view the model as a rough starting point for multinomial data, analogous to using ordinary least squares for continuous data, and obtain cluster-robust standard errors. For an unordered m alternative random-effects (RE) model, we can introduce $(m - 1)$ normally distributed cluster-specific random intercepts $\alpha_{g2}, \dots, \alpha_{gm}$, where we normalize $\alpha_1 = 0$. These are then integrated out using numerical methods. This RE estimator is implemented by the `xtmlogit` and `cmxtmixlogit` commands.

For an unordered multinomial model that has cluster-specific fixed effects, the CL method for binary choice can be extended to the MNL model. The community-contributed `femlogit` command implements this estimator; see [Pforr \(2014\)](#).

For ordered models based on a single latent variable crossing a threshold, a RE model needs to simply introduce a single normally distributed random intercept α_g . This model can be fit using the `xtologit` or `xtoprobit` command. The multilevel mixed-effects `meologit` or `meoprobit` command also fit this RE model and, additionally, can allow for cluster-specific random-slope coefficients.

Consistency of resulting parameter estimates for all of these methods requires the assumption that individual observations be independent within cluster once cluster-specific effects are included. So even though there is a

`vce(robust)` option, if it needs to be used, then the parameter estimates are most likely inconsistent.

18.11 Multivariate outcomes

We consider the multinomial analog of the seemingly unrelated regression (SUR) model (see section 6.8), where two or more categorical outcomes are being modeled.

In the simplest case, outcomes do not directly depend on each other—there is no simultaneity, but the errors for the outcomes may be correlated. When the errors are correlated, a more efficient estimator that models the joint distribution of the errors is available.

In more complicated cases, the outcomes depend directly on each other, so there is simultaneity. We do not cover this case, but analysis is much simpler if the simultaneity is in continuous latent variables rather than discrete outcome variables.

18.11.1 Bivariate probit

The bivariate probit model considers two binary outcomes. The outcomes are potentially related after conditioning on regressors. The relatedness occurs via correlation of the errors that appear in the index-function model formulation of the binary outcome model.

Specifically, the two outcomes are determined by two unobserved latent variables,

$$\begin{aligned}y_1^* &= \mathbf{x}'_1 \boldsymbol{\beta}_1 + \varepsilon_1 \\y_2^* &= \mathbf{x}'_2 \boldsymbol{\beta}_2 + \varepsilon_2\end{aligned}$$

where the errors ε_1 and ε_2 are jointly normally distributed with means of 0, variances of 1, and correlations of ρ , and we observe the two binary outcomes

$$y_1 = \begin{cases} 1 & \text{if } y_1^* > 0 \\ 0 & \text{if } y_1^* \leq 0 \end{cases} \quad \text{and} \quad y_2 = \begin{cases} 1 & \text{if } y_2^* > 0 \\ 0 & \text{if } y_2^* \leq 0 \end{cases}$$

The model collapses to two separate probit models for y_1 and y_2 if $\rho = 0$.

There are four mutually exclusive outcomes that we can denote by y_{10} (when $y_1 = 1$ and $y_2 = 0$), y_{01} , y_{11} , and y_{00} . The log-likelihood function is derived using the expressions for these probabilities, and the parameters are estimated by ML. There are two complications. First, there is no analytical expression for the probabilities, because they depend on a one-dimensional integral with no closed-form solution, but this is easily solved with numerical quadrature methods for integration. Second, the resulting expressions for $\Pr(y_1 = 1|x)$ and $\Pr(y_2 = 1|x)$ differ from those for binary probit and probit.

The simplest form of the bivariate command has the syntax

```
biprobit depvar1 depvar2 [indepvars] [if] [in] [weight] [, options]
```

This version assumes that the same regressors are used for both outcomes. A more general version allows the list of regressors to differ for the two outcomes.

We consider two binary outcomes using the same dataset as that for ordered outcome models analyzed in section [18.9](#). The first outcome is the `hlthe` variable, which takes a value of 1 if self-assessed health is excellent and 0 otherwise. The second outcome is the `dmdu` variable, which equals 1 if the individual has visited the doctor in the past year and 0 otherwise. A data summary is

```
. * Two binary dependent variables: hlthe and dmdu  
. tabulate hlthe dmdu
```

hlthe	Any MD visit; 1 if mdu>0		Total
	0	1	
0	826	1,731	2,557
1	1,006	2,011	3,017
Total	1,832	3,742	5,574

	hlthe	dmdu
hlthe	1.0000	
dmdu	-0.0110	1.0000

The outcomes are very weakly negatively correlated, so in this case, there may be little need to model the two jointly.

Bivariate probit model estimation yields the following estimates:

```
. * Bivariate probit estimates
. biprobit hlthe dmdu age linc ndisease, nolog vce(robust)

Bivariate probit regression
Number of obs = 5,574
Wald chi2(6) = 767.89
Prob > chi2 = 0.0000
Log pseudolikelihood = -6958.0751
```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
hlthe	age	-.0178246	.0010632	-16.77	0.000	-.0199084 -.0157407
	linc	.132468	.0164203	8.07	0.000	.1002847 .1646513
	ndisease	-.0326656	.0027701	-11.79	0.000	-.038095 -.0272362
	_cons	-.2297079	.1462127	-1.57	0.116	-.5162794 .0568637
dmdu	age	.0020038	.0010748	1.86	0.062	-.0001028 .0041103
	linc	.1212519	.0155242	7.81	0.000	.090825 .1516788
	ndisease	.0347111	.0029126	11.92	0.000	.0290025 .0404198
	_cons	-1.032527	.1401069	-7.37	0.000	-1.307132 -.7579229
/athrho	.0282258	.022842	1.24	0.217	-.0165437	.0729953
rho	.0282183	.0228238			-.0165422	.0728659

Wald test of rho=0: chi2(1) = 1.52695 Prob > chi2 = 0.2166

The hypothesis that $\rho = 0$ is not rejected, so in this case, bivariate probit was not necessary. As might be expected, separate probit estimation for each outcome (output not given) yields coefficients very similar to those given above.

Predicted probabilities can be obtained. For example, the marginal probability that $y_1 = 1$ can be obtained with the `pmarg1` option, whereas the joint probability that $(y_1, y_2) = (1, 1)$ is obtained with the `p11` option. We obtain

```

. * Predicted probabilities
. predict biprob1, pmarg1
. predict biprob2, pmarg2
. predict biprob11, p11
. predict biprob10, p10
. predict biprob01, p01
. predict biprob00, p00
. summarize hlthe dmdu biprob1 biprob2 biprob11 biprob10 biprob01 biprob00

```

Variable	Obs	Mean	Std. dev.	Min	Max
hlthe	5,574	.541263	.4983392	0	1
dmdu	5,574	.6713312	.4697715	0	1
biprob1	5,574	.5414237	.1577588	.0156161	.7853771
biprob2	5,574	.6716857	.0976294	.1589158	.9834746
biprob11	5,574	.3610553	.0989285	.0090629	.5492701
biprob10	5,574	.1803685	.0765047	.0006476	.3680022
biprob01	5,574	.3106305	.1434517	.1090853	.9385432
biprob00	5,574	.1479458	.064902	.0158778	.6909308

The marginal probabilities that $y_1 = 1$ and $y_2 = 1$ are, respectively, 0.541 and 0.672, very close to the sample frequencies.

18.11.2 Nonlinear SUR

An alternative model is to use the `nlsur` command for nonlinear SUR, where the conditional mean of y_1 is $\Phi(\mathbf{x}'_1 \boldsymbol{\beta}_1)$ and of y_2 is $\Phi(\mathbf{x}'_2 \boldsymbol{\beta}_2)$. This estimator does not control for the intrinsic heteroskedasticity of binary outcome data, so we use the `vce(robust)` option to obtain standard errors that control for both heteroskedasticity and correlation. We have

```

. * Nonlinear seemingly unrelated regressions estimator
. nlsur (hlthe = normal({a1}*age+{a2}
>      *linc+{a3}*ndisease+{a4}))
>      (dmdu = normal({b1}*age+{b2}*linc+{b3}
>      *ndisease+{b4})), vce(robust) nolog
(obs = 5,574)
Calculating NLS estimates...
Calculating FGNLS estimates...
FGNLS regression

```

Equation		Obs	Parms	RMSE	R-sq	Constant
1	hlthe	5,574	4	.4727309	0.5871*	(none)
2	dmdu	5,574	4	.4595438	0.6854*	(none)

* Uncentered R-sq

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
/a1	-.0173125	.0010624	-16.30	0.000	-.0193948 -.0152302
/a2	.1486604	.0184521	8.06	0.000	.1124949 .1848259
/a3	-.0333346	.0028682	-11.62	0.000	-.0389562 -.027713
/a4	-.3790899	.1638203	-2.31	0.021	-.7001719 -.0580079
/b1	.0018343	.0010776	1.70	0.089	-.0002778 .0039464
/b2	.1270039	.0165602	7.67	0.000	.0945465 .1594614
/b3	.0345088	.0030258	11.40	0.000	.0285783 .0404393
/b4	-1.081392	.1496894	-7.22	0.000	-1.374778 -.788006

For this example, the regression coefficients and standard errors are quite similar to those from `biprobit`.

18.12 Additional resources

The key models for initial understanding are the MNL and CL models. In practice, these models are often too restrictive. Stata commands cover most multinomial models. [Train \(2009\)](#) is an excellent source, especially for models that need to be fit by MSL or Bayesian methods.

18.13 Exercises

1. Consider the health-status multinomial example of section [18.9](#). Refit this as a MNL model using the `mlogit` command. Comment on the statistical significance of regressors. Obtain the MES of changes in the regressors on the probability of excellent health for the MNL model, and compare these to those given in section [18.9.5](#) for the ordered logit model. Using Bayesian information criterion, which model do you prefer for these data—MNL or ordered logit?
2. Consider the CL example of section [18.5](#). Use `mus218hk.dta`, if necessary to create this file as in section [18.5.1](#). Drop the charter boat option as in section [18.7.5](#), using `drop if fishmode=="charter" | mode==4`, so we have a three-choice model. Estimate the parameters of a CL model with regressors `p` and `q` and income, using the `cmclogit` command. What are the MES on the probability of private boat fishing of a \$10 increase in the price of private boat fishing, a one-unit change in the catch rate from private boat fishing, and a \$1,000 increase in monthly income? Which model fits these data better—the CL model of this question or the MNP model of section [18.7](#)?
3. Continue the previous question, a three-choice model for fishing mode. Estimate the parameters of the model by NL, with errors for the utility of pier and beach fishing correlated with each other and uncorrelated with the error for the utility of private boat fishing. Obtain the ME of a change in the price of private boat fishing, adapting the example of section [18.6.6](#).
4. Continue with the fishing data example of section [18.5](#), with `beach` as the base alternative and `income` as the case variable. Instead of fitting an alternative-specific conditional logit model, fit the random-parameter logit model of section [18.8](#) using the `cmmixlogit` command. Is this model a significant improvement on the fixed parameter model? Use the `margins` command to fit the ME of income on the choice of fishing mode. Compare the role of income in this model with that in the model of section [18.5](#).
5. Consider the health-status multinomial example of section [18.9](#). Estimate the parameters of this model as an ordered probit model using the `oprobit` command. Comment on the statistical significance of

regressors. Obtain the MES for the predicted probability of excellent health for the ordered probit model, and compare these to those given in section [18.9.5](#) for the ordered logit model. Which model do you prefer for these data—ordered probit or ordered logit?

Chapter 19

Tobit and selection models

19.1 Introduction

The tobit model and its generalizations are relevant when the dependent variable of a linear regression is incompletely observed.

The leading example is censoring, where for some observations the dependent variable is observed only over some interval of its support. A simple example is top-coding of outcomes such as annual income. A more subtle example is modeling desired annual expenditures on a new automobile, in which case a cross-sectional survey will reveal a significant proportion of households with zero expenditure and the rest with a positive level of expenditure.

A related example is truncation, where some observations are not observed if they fall outside some interval of support. For example, we may observe only those individuals with income less than the top code or observe only those individuals with positive annual expenditures on a new car.

Even if the underlying model is linear, ordinary least-squares (OLS) regression will not yield consistent parameter estimates, because a censored sample or truncated sample is not representative of the population. Instead, alternative estimation methods are needed.

We begin with the basic tobit model, which assumes that the underlying regression model is a latent variable model with dependent variable that follows a linear regression model with independent homoskedastic normally distributed errors. This latent variable is only partially observed.

We then consider extensions, including models for a dependent variable in logarithms, a two-part model in both of which the mechanism generating zero outcome has a special role, a particular selection model (the “Heckman model”) that allows errors in the two parts to be correlated, and models with endogenous regressors. Note, however, that consistency of estimators relies very much on normality of errors or minor relaxation of this assumption. To extrapolate from a censored sample to an underlying population requires strong assumptions.

Finally, the methods developed in this chapter are used to compare balanced and unbalanced samples resulting from panel attrition and to correct for attrition bias in linear panel models.

19.2 Tobit model

Suppose that our data consist of (y_i, \mathbf{x}_i) , $i = 1, \dots, N$. Assume that \mathbf{x}_i is fully observed but y_i is incompletely observed.

19.2.1 Regression with censored data

Suppose the household has a latent (unobserved) demand for goods, denoted by y^* , that is not expressed as a purchase until some known constant threshold, denoted by L , is passed. We observe $y = y^*$ only when $y^* > L$; otherwise, we observe $y = 0$. This is an example of left-censoring or censoring from below.

If instead we observe $y = y^*$ only when $y^* < U$, and otherwise observe $y = U$, then we have right-censoring or censoring from above.

This section focuses on left-censoring or censoring from below because this is the most common form of censoring with individual level economics data.

19.2.2 Tobit model setup

The regression of interest is specified as an unobserved latent variable, y^* ,

$$y_i^* = \mathbf{x}'_i \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, \dots, N \tag{19.1}$$

where $\varepsilon_i \sim N(0, \sigma^2)$ and \mathbf{x}_i denotes the $(K \times 1)$ vector of exogenous and fully observed regressors. If y^* were observed, we would estimate $(\boldsymbol{\beta}, \sigma^2)$ by OLS in the usual way.

The observed variable y_i is related to the latent variable y_i^* through the observation rule

$$y = \begin{cases} y^* & \text{if } y^* > L \\ L & \text{if } y^* \leq L \end{cases} \tag{19.2}$$

The probability of an observation being censored is

$\Pr(y^* \leq L) = \Pr(\mathbf{x}'_i \boldsymbol{\beta} + \varepsilon \leq L) = \Phi \{ (L - \mathbf{x}'_i \boldsymbol{\beta})/\sigma \}$, where $\Phi(\cdot)$ is the standard normal cumulative distribution function.

The truncated mean, or expected value, of y for the noncensored observations can be shown to be

$$E(y_i | \mathbf{x}_i, y_i > L) = \mathbf{x}'_i \boldsymbol{\beta} + \sigma \frac{\phi \{ (\mathbf{x}'_i \boldsymbol{\beta} - L)/\sigma \}}{\Phi \{ (\mathbf{x}'_i \boldsymbol{\beta} - L)/\sigma \}} \quad (19.3)$$

where $\phi(\cdot)$ is the standard normal density and it is assumed that $\varepsilon_i \sim N(0, \sigma^2)$. The conditional mean $E(y_i | \mathbf{x}_i, y_i > L)$ in (19.3) differs from $\mathbf{x}'_i \boldsymbol{\beta}$, as does the censored mean $E(y_i | \mathbf{x}_i)$ which, for example, equals $\Phi(\mathbf{x}'_i \boldsymbol{\beta}/\sigma) \mathbf{x}'_i \boldsymbol{\beta} + \phi(\mathbf{x}'_i \boldsymbol{\beta}/\sigma)$ when $L = 0$. Thus, OLS of y on \mathbf{x} leads to an inconsistent estimate of $\boldsymbol{\beta}$ in the original model (19.1).

A sample may instead include right-censored observations. Then we observe that

$$y = \begin{cases} y^* & \text{if } y^* < U \\ U & \text{if } y^* \geq U \end{cases}$$

The leading case of censoring is that in which the data are left-censored only and $L = 0$. A variant of the tobit model, the two-limit tobit, allows for both left- and right-censoring. Another variant, considered in the application of this chapter, is that in which the data are only left-censored but L is unknown.

19.2.3 Tobit estimation

The tobit model can be fit by maximum likelihood (ML) or by two-step regression. We first consider ML estimation under the assumptions that the regression error is homoskedastic and normally distributed.

For the case of left-censored data with censoring point L , the density function has two components that correspond to uncensored and censored observations. Let $d = 1$ denote the censoring indicator for the outcome that the observation is not censored, and let $d = 0$ indicate a censored observation. The density can be written as

$$f(y_i) = \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{1}{2\sigma^2} (y_i - \mathbf{x}'_i \boldsymbol{\beta})^2 \right\} \right]^{d_i} [\Phi\{(L - \mathbf{x}'_i \boldsymbol{\beta})/\sigma\}]^{1-d_i} \quad (19.4)$$

The second term in (19.4) reflects the contribution to the likelihood of the censored observation. ML estimates of $(\boldsymbol{\beta}, \sigma^2)$ solve the first-order conditions from maximization of the log likelihood based on (19.4). These equations are nonlinear in parameters, so an iterative algorithm is required.

An alternative two-step estimator requires slightly weaker assumptions than those for the tobit maximum-likelihood estimator (MLE). Suppose we consider regression using only the uncensored observations. Then the conditional mean in (19.3) has an additional term ϕ_i/Φ_i . This missing variable can be generated by a probit model that models the probability of the outcome that $y_i^* > 0$. Let $d_i = 1$ denote the outcome that $y_i^* > 0$, and let $d_i = 0$ otherwise. Probit estimation of d_i on \mathbf{x}_i using all observations can yield a consistent estimate of $\boldsymbol{\beta}$ and hence a consistent estimate of $\lambda_i = \phi_i/\Phi_i$. A linear regression of y_i on \mathbf{x} and $\widehat{\phi_i/\Phi_i}$ using just the uncensored observations will provide a consistent estimate of $\boldsymbol{\beta}$.

19.2.4 Robust standard errors

The tobit MLE is consistent under the stated assumptions, including $\varepsilon_i \sim N(0, \sigma^2)$. It is inconsistent, however, if the errors are not normally distributed or if they are heteroskedastic. These strong assumptions are likely to be violated in applications, and this makes the tobit MLE a nonrobust estimator. It is desirable to test the assumptions of normality and heteroskedasticity.

Some misspecification is likely, so inference should be based on quasi-ML theory; see section 13.3.1. We use robust standard errors that provide

consistent estimates of the variance of $\hat{\beta}$, should the model be misspecified. For independent observations, we use the `vce(robust)` option, and for observations that are dependent because of clustering, we use the `vce(cluster clustvar)` option.

19.2.5 The tobit command

The Stata estimation command for tobit regression with censored data has the following basic syntax:

```
tobit depvar [indepvars] [if] [in] [weight] [, ll[(varname|#)] ul[(varname|#)]  
options]
```

The specifications `ll[(#)]` and `ul[(#)]` refer to the lower limit (left-censoring point) and the upper limit (right-censoring point), respectively. If the data are subject to left-censoring at zero, for example, then only `ll(0)` is required. Similarly, only `ul(10000)` is required for right-censored data at the censoring point 10,000. Both are required if the data are both right-censored and left-censored and if one wants to estimate the parameters of the two-limit tobit model. The postestimation tools for `tobit` will be discussed later in this chapter.

19.2.6 Unknown censoring point

As [Carson and Sun \(2007\)](#), and others, have pointed out, the censoring point may be unknown.

Suppose that the data are left-censored, with a constant but unknown threshold, γ , in (19.2). The commonly used assumption that the unknown γ can be set to zero as a “normalization” is not innocuous. Instead, $\Pr(y^* < \gamma) = \Phi\{(\gamma - \mathbf{x}'\boldsymbol{\beta})/\sigma\}$, where $(\gamma - \mathbf{x}'\boldsymbol{\beta})/\sigma$ is interpreted as a “threshold”. In this case, we can set $\hat{\gamma} = \min(\text{uncensored } y)$ and proceed as if γ is known. Estimates of the tobit model based on this procedure have been shown to be consistent; see [Carson and Sun \(2007\)](#).

In Stata, this requires only that the value of γ should be used in defining the lower limit, so we can again use the `tobit` command with the `ll(#)` option. It is simplest to set # equal to $\hat{\gamma}$. However, this will treat the

observation or observations with $y = \hat{\gamma}$ as censored, and a better alternative is to set $\#$ equal to $\hat{\gamma} - \Delta$ for some small value, Δ , such as 10^{-6} .

19.3 Tobit model example

The illustration we consider, ambulatory expenditures, has the very common complication that the data are highly right skewed. This is best treated by taking the natural logarithm, complicating the analysis of these complications.

In the current section, we instead present the simpler tobit model in levels. The subsequent sections [19.4–19.8](#) present the tobit model in natural logarithms and consequent complications. Model diagnostics are presented in section [19.4.5](#).

19.3.1 Data summary

The data on the dependent variable for ambulatory expenditure (`ambexp`) and the regressors (`age`, `female`, `educ`, `blhisp`, `totchr`, and `ins`) are taken from the 2001 Medical Expenditure Panel Survey and are for people aged 21–64 years. In this sample of 3,328 observations, there are 526 (15.8%) 0 values of `ambexp`, so censoring may be an issue.

Descriptive statistics for all the variables follow:

```
. * Raw data summary  
. qui use mus219mepsambexp, clear  
. summarize ambexp age female educ blhisp totchr ins
```

Variable	Obs	Mean	Std. dev.	Min	Max
ambexp	3,328	1386.519	2530.406	0	49960
age	3,328	4.056881	1.121212	2.1	6.4
female	3,328	.5084135	.5000043	0	1
educ	3,328	13.40565	2.574199	0	17
blhisp	3,328	.3085938	.4619824	0	1
totchr	3,328	.4831731	.7720426	0	5
ins	3,328	.3650841	.4815261	0	1

A detailed summary of `ambexp` provides insight into the potential problems in estimating the parameters of the tobit model with a linear conditional mean function.

```
. * Detailed summary to show skewness and kurtosis
. summarize ambexp, detail
```

Annual ambulatory expenditures

	Percentiles	Smallest		
1%	0	0		
5%	0	0		
10%	0	0	Obs	3,328
25%	113	0	Sum of wgt.	3,328
50%	534.5		Mean	1386.519
		Largest	Std. dev.	2530.406
75%	1618	28269		
90%	3585	30920	Variance	6402953
95%	5451	34964	Skewness	6.059491
99%	11985	49960	Kurtosis	72.06738

The `ambexp` variable is very heavily skewed and has considerable nonnormal kurtosis. This feature of the dependent variable should alert us to the possibility that the tobit MLE may be a flawed estimator for the model.

To see whether these characteristics persist if the zero observations are ignored, we examine the sample distribution of only positive values.

```
. * Summary for positives only
. summarize ambexp if ambexp >0, detail
```

Annual ambulatory expenditures

	Percentiles	Smallest		
1%	22	1		
5%	67	2		
10%	107	2	Obs	2,802
25%	275	4	Sum of wgt.	2,802
50%	779		Mean	1646.8
		Largest	Std. dev.	2678.914
75%	1913	28269		
90%	3967	30920	Variance	7176579
95%	6027	34964	Skewness	5.799312
99%	12467	49960	Kurtosis	65.81969

The skewness and nonnormal kurtosis are reduced only a little if the zeros are ignored.

In principle, the skewness and nonnormal kurtosis of `ambexp` could be due to regressors that are skewed. But, from output not listed, an OLS

regression of `ambexp` on `age`, `female`, `educ`, `blhisp`, `totchr`, and `ins` explains little of the variation ($R^2 = 0.16$), and the OLS residuals have a skewness statistic of 6.6 and a kurtosis statistic of 92.2. Even after conditioning on regressors, the dependent variable is very nonnormal, and a lognormal model may be more appropriate.

19.3.2 Tobit analysis

As an initial exploratory step, we will run the linear tobit model without any transformation of the dependent variable, even though it appears that the data distribution may be nonnormal. The lowest positive values of `ambexp` (of 1, 2, 2, and 4) are close to zero compared with the median of positive values of 779, so it is natural to set the left-censoring point L at 0 in this example.

```

. * Tobit analysis for ambexp using all expenditures
. global xlist age female educ blhisb totchr ins // Define regressor list $xlist
. tobit ambexp $xlist, ll(0) vce(robust)

Refining starting values:
Grid node 0: log likelihood = -26432.72

Fitting full model:
Iteration 0: log pseudolikelihood = -26432.72
Iteration 1: log pseudolikelihood = -26361.187
Iteration 2: log pseudolikelihood = -26359.43
Iteration 3: log pseudolikelihood = -26359.424
Iteration 4: log pseudolikelihood = -26359.424

Tobit regression
Number of obs      = 3,328
Uncensored          = 2,802
Left-censored       = 526
Right-censored      = 0
F(6, 3322)          = 59.52
Prob > F            = 0.0000
Pseudo R2           = 0.0130

Log pseudolikelihood = -26359.424

```

ambexp	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
age	314.1479	41.19122	7.63	0.000	233.3852	394.9107
female	684.9918	100.1353	6.84	0.000	488.6585	881.325
educ	70.8656	17.25925	4.11	0.000	37.02577	104.7054
blhisb	-530.311	102.8097	-5.16	0.000	-731.8877	-328.7342
totchr	1244.578	98.91188	12.58	0.000	1050.644	1438.513
ins	-167.4714	84.42021	-1.98	0.047	-332.9923	-1.95054
_cons	-1882.591	317.2026	-5.93	0.000	-2504.524	-1260.659
var(e.ambexp)	6635296	1088362			4810499	9152305

All regressors are statistically significant at the 0.05 level. The interpretation of the coefficients is as a partial derivative of the latent variable, y^* , with respect to x . Marginal effects (MES) for the observed variable, y , are presented in section [19.3.4](#).

19.3.3 Prediction after tobit

The `predict` command is summarized in section 4.2. The command can be used after `tobit` to predict a range of quantities. We begin with the default linear prediction, `xb`, that produces fitted values of the latent variable, y^* , for all observations in the sample.

```

. * Tobit prediction and summary
. predict yhatlin
(option xb assumed; fitted values)
. summarize yhatlin

```

Variable	Obs	Mean	Std. dev.	Min	Max
yhatlin	3,328	1066.683	1257.455	-1564.703	8027.957

A more detailed comparison of the sample statistics for `yhatlin` with those for `ambexp` shows that the tobit model fits especially poorly in the upper tail of the distribution.

This fact notwithstanding, we will use the model to illustrate other prediction options for the observed variable, y , that can be used in combination with the computation of MES.

19.3.4 MEs

In a censored regression, there are a variety of MES that are of potential interest; see [R] **tobit postestimation**. The ME is the effect on the conditional mean of the dependent variable of changes in the regressors. This effect varies according to whether interest lies in the latent variable mean, the truncated mean or the censored mean.

Omitting derivations given in [Cameron and Trivedi \(2005\)](#), chap. 16), for censoring from below at zero and with censoring observations taking value zero, we see the MES are

Latent variable	$\partial E(y^* \mathbf{x})/\partial \mathbf{x} = \boldsymbol{\beta}$
Left-truncated (at 0)	$\partial E(y \mathbf{x}, y > 0)/\partial \mathbf{x} = \{1 - w\lambda(w) - \lambda(w)^2\}\boldsymbol{\beta}$
Left-censored (at 0)	$\partial E(y \mathbf{x})/\partial \mathbf{x} = \Phi(w)\boldsymbol{\beta}$

where $w = \mathbf{x}'\boldsymbol{\beta}/\sigma$ and $\lambda(w) = \phi(w)/\Phi(w)$. The first of these two has already been discussed above.

Left-truncated and left-censored examples

The `predict()` option of `margins` is used to obtain MES with respect to a desired quantity such as the left-truncated mean. Here we obtain the ME at the mean value of the regressors (MEM), using the `atmeans` option, so the MES are for the average individual in the sample. Factor-variable notation is used so that the MEMs for indicator variables are computed using the finite-difference method.

We begin with the MEMs for the left-truncated mean, $E(y|\mathbf{x}, y > 0)$.

```
. * (1) MEMs for E(y|x,y>0)
. qui tobit ambexp age i.female educ i.blhisp totchr i.ins, ll(0) vce(robust)
. margins, dydx(*) predict(e(0,.)) atmeans noatlegend
Conditional marginal effects                                         Number of obs = 3,328
Model VCE: Robust
Expression: E(ambexp|ambexp>0), predict(e(0,.))
dy/dx wrt:  age 1.female educ 1.blhisp totchr 1.ins
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
age	145.524	18.79351	7.74	0.000	108.6893	182.3586
1.female	317.1037	44.11704	7.19	0.000	230.6359	403.5715
educ	32.82734	7.790956	4.21	0.000	17.55735	48.09733
1.blhisp	-240.2953	46.5901	-5.16	0.000	-331.6102	-148.9804
totchr	576.5307	44.95038	12.83	0.000	488.4296	664.6318
1.ins	-77.19554	38.28773	-2.02	0.044	-152.2381	-2.152964

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

For example, for the average individual 10 more years of aging (a 1-unit change in variable `age`) is associated with a \$146 increase in expenditures, conditional on his or her expenditures being positive. For these data, the MEMs are roughly one-half of the coefficient estimates, $\hat{\beta}$, given in section [19.3.2](#).

The MEMs for the censored mean, $E(y|\mathbf{x})$, are computed next.

```

. * (2) MEMs for E(y|x)
. margins, dydx(*) predict(ystar(0,.)) atmeans noatlegend
Conditional marginal effects                                         Number of obs = 3,328
Model VCE: Robust
Expression: E(ambexp*|ambexp>0), predict(ystar(0,.))
dy/dx wrt:  age 1.female educ 1.blhisp totchr 1.ins

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
age	207.526	26.80233	7.74	0.000	154.9944	260.0576
1.female	451.6399	62.75074	7.20	0.000	328.6507	574.6291
educ	46.81378	11.11639	4.21	0.000	25.02605	68.60151
1.blhisp	-342.4803	66.2929	-5.17	0.000	-472.412	-212.5486
totchr	822.1678	64.07788	12.83	0.000	696.5774	947.7581
1.ins	-110.0883	54.6086	-2.02	0.044	-217.1192	-3.05739

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

For example, it is predicted that regardless of whether medical expenditures are positive, for the average individual 1 more year of aging is associated with a \$208 increase in expenditures. These MEMs for the average individual are larger in absolute value than those for the left-truncated mean and are roughly 70% of the original coefficient estimates.

Left-censored case computed directly

Next, we illustrate direct computation of the MES for the left-censored mean, which we see above is $\Phi(\bar{x}'\hat{\beta}/\hat{\sigma})\hat{\beta}_j$ for the j th regressor.

This example also illustrates how to retrieve tobit model coefficients. There are six regressors, but three are binary indicator variables that each take up two components in $e(b)$. So the first 9 ($= 3 + 2 \times 3$) components of $e(b)$ are regressors, the 10th is the constant term, and the 11th is the estimate of σ^2 . We have

```

. * Direct computation of MEMs for E(y|x)
. predict xb1, xb           // xb1 is estimate of x'b
. matrix btobit = e(b)
. scalar sigmasq = btobit[1,11] // sigmasq is estimate of sigma^2
. matrix bcoeff = btobit[1,1..9] // bcoeff is betas excl. constant
. qui summarize xb1
. scalar meanxb = r(mean)      // Mean of x'b equals (mean of x)'b
. scalar PHI = normal(meanxb / sqrt(sigmasq))
. matrix deriv = PHI*bcoeff

. matrix list deriv
deriv[1,9]
      ambexp:    ambexp:    ambexp:    ambexp:    ambexp:    ambexp:
                  0b.        1.          0b.        1.
      age       female     female     educ      blhisp     blhisp
y1   207.52598          0   452.50523   46.813781          0  -350.32317
      ambexp:    ambexp:    ambexp:
                  0b.        1.
      totchr     ins       ins
y1   822.16778          0  -110.63154

. ereturn post deriv          // Print nicer looking results
. ereturn display

```

	Coefficient
ambexp	
age	207.526
1.female	452.5052
educ	46.81378
1.blhisp	-350.3232
totchr	822.1678
1.ins	-110.6315

As expected, the MEMs for continuous regressors are identical to those obtained above with `margins`. For binary regressors, there is some difference because `margins` uses the finite-difference method rather than calculus methods; see section 13.7.6.

Marginal impact on probabilities

Interest may also lie in the impact of a change in a regressor on the probability that y is in a specified interval. For illustration, consider the MEMs for $\text{Pr}(5000 < \text{ambexp} < 10000)$.

```

. * Compute MEMs for Pr(5000<ambexp<10000)
. qui tobit ambexp age i.female educ i.blhisp totchr i.ins, ll(0) vce(robust)
. margins, dydx(*) predict(pr(5000,10000)) atmeans noatlegend
Conditional marginal effects                                         Number of obs = 3,328
Model VCE: Robust
Expression: Pr(5000<ambexp<10000), predict(pr(5000,10000))
dy/dx wrt:  age 1.female educ 1.blhisp totchr 1.ins

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
age	.0150449	.002801	5.37	0.000	.0095551	.0205347
1.female	.0328152	.0068034	4.82	0.000	.0194808	.0461496
educ	.0033938	.0009929	3.42	0.001	.0014478	.0053399
1.blhisp	-.0239676	.0049076	-4.88	0.000	-.0335864	-.0143488
totchr	.0596042	.0090396	6.59	0.000	.0418869	.0773214
1.ins	-.0079162	.0043282	-1.83	0.067	-.0163994	.0005669

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

The effects appear very small, though note that from separate computation, only 5% of the sample falls into this range.

19.3.5 Truncated regression

In some applications, the data are truncated, so that only data that are not censored are observed. In that case, the truncated tobit MLE can be obtained using the `truncreg` command.

For expositional purposes, we can generate truncated data by restricting attention to only those observations with `ambexp > 0`. The truncated tobit estimates with left-truncation at zero are then

```

. * Truncated tobit with truncation at zero
. truncreg ambexp age female educ blhisp totchr ins, ll(0) nolog vce(robust)
(526 obs truncated)

Truncated regression
Limit: Lower =      0                               Number of obs      =     2,802
                    Upper = +inf                         Wald chi2(6)      =      7.58
Log pseudolikelihood = -23265.704                  Prob > chi2       =     0.2706

```

ambexp	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
age	28466.17	13473.91	2.11	0.035	2057.793	54874.55
female	34695.92	22217.13	1.56	0.118	-8848.853	78240.69
educ	1226.796	1909.361	0.64	0.521	-2515.483	4969.076
blhisp	-33156.71	15953.57	-2.08	0.038	-64425.13	-1888.297
totchr	42830.02	18250.76	2.35	0.019	7059.18	78600.85
ins	-26356.62	16758.18	-1.57	0.116	-59202.05	6488.81
_cons	-371878.7	181883.8	-2.04	0.041	-728364.5	-15392.94
/sigma	17530.02	4362.635	4.02	0.000	8979.409	26080.62

The coefficients have the same sign as those from censored tobit or OLS regression but are many times larger and appear to be unrealistic. For example, 10 years of aging is estimated to lead to \$28,466 higher ambulatory expenditures.

However, parameter interpretation can become difficult in nonlinear models, and it is MMEs that matter. For example, the MMEs for the left-truncated mean, $E(y|x, y > 0)$, are

```

. * MMEs for E(y|x) following truncated tobit
. margins, dydx(*) predict(e(0,.)) atmeans noatlegend
Conditional marginal effects                                         Number of obs = 2,802
Model VCE: Robust
Expression: E(ambexp|ambexp>0), predict(e(0,.))
dy/dx wrt:  age female educ blhisp totchr ins

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
age	184.017	27.3567	6.73	0.000	130.3988	237.6351
female	224.2886	72.98036	3.07	0.002	81.24976	367.3275
educ	7.930513	11.56309	0.69	0.493	-14.73274	30.59376
blhisp	-214.3386	95.91943	-2.23	0.025	-402.3372	-26.33997
totchr	276.8708	23.0198	12.03	0.000	231.7528	321.9888
ins	-170.38	63.4146	-2.69	0.007	-294.6703	-46.08967

These are much more comparable with the corresponding MEMS following censored tobit estimation.

19.3.6 Additional commands for censored and truncated regression

The censored least-absolute-deviations estimator of [Powell \(1984\)](#) provides consistent estimates for left-censored or right-censored data under the weaker assumption that the error, ε , in (19.1) is independent and identically distributed and symmetrically distributed. This is implemented with the community-contributed `clad` command of [Jolliffe, Krushelnitsky, and Semykina \(2000\)](#). For these data, the method is best implemented for the data in logs.

The `intreg` command is a generalization of `tobit` for data observed in intervals. For example, expenditures might be observed in the ranges $y \leq 0$, $0 < y \leq 1000$, $1000 < y \leq 10000$, and $y \geq 10000$.

A quite different type of right-censored data is duration data on length of unemployment spell or survival data on time until death. The standard approach for such data is to model the conditional hazard of the spell ending rather than the conditional mean. Modeling the conditional hazard has the advantage of permitting the use of the Cox proportional hazards model, which allows semiparametric estimation without strong distributional assumptions such as an exponential or Weibull distribution for durations. For details, see section [21.5](#).

For ML estimates of censored Poisson and truncated Poisson and negative binomial for count data, see section [20.3.7](#).

The `gsem` command presented in section [23.6](#) enables estimation of censored regression for a normally distributed dependent variable and truncated regression for normally distributed, Poisson, and several parametric survival models.

The following command gives results identical to those obtained earlier using the `tobit` command.

```
. * gsem alternative for estimating tobit regression
. gsem ambexp <- $xlist, family(gaussian, lensored(0)) vce(robust)
(output omitted)
```

For any given latent variable density $f^*(y^*|\mathbf{x}, \boldsymbol{\theta})$, it is easy to code up the log density for censored or truncated y and obtain ML estimates using the `mlexp` or `ml` command. For example, for y_i left-truncated at zero, the log density is

$$\ln f(y_i|\mathbf{x}_i, \boldsymbol{\theta}) = d_i \ln f^*(y_i|\mathbf{x}_i, \boldsymbol{\theta}) + (1 - d_i) \ln F^*(0_i|\mathbf{x}_i, \boldsymbol{\theta})$$

where $d_i = 1$ if $y_i > 0$, $d_i = 0$ if $y_i = 0$, and $F^*(y^*|\mathbf{x}, \boldsymbol{\theta})$ is the cumulative distribution function of y^* .

19.3.7 Clustered data

The tobit model is highly dependent on distributional assumptions and not robust to within-cluster correlation. Instead, one needs to generalize the tobit model to a richer parametric model that explicitly incorporates within-clustering correlation.

The `metobit` command allows a normally distributed intercept and slope parameters to vary at the cluster level, as does the `meintreg` command for interval data; see section [23.4](#). For panel data, the `xttobit` and `xtintreg` commands allow the intercept to vary at the cluster level.

19.4 Tobit for lognormal data

The tobit model relies crucially on normality, but expenditure data are often better modeled as lognormal. A tobit regression model for lognormal data introduces two complications: a nonzero threshold and lognormal y .

Now introduce lognormality by specifying

$$y^* = \exp(\mathbf{x}'\boldsymbol{\beta} + \varepsilon), \quad \varepsilon \sim N(0, \sigma^2)$$

where we observe that

$$y = \begin{cases} y^* & \text{if } \ln y^* > \gamma \\ 0 & \text{if } \ln y^* \leq \gamma \end{cases}$$

Here it is known that $y = 0$ when data are censored, and in general $\gamma \neq 0$. The parameters of this model can be fit using the `tobit` command with the `ll(#)` option, where the dependent variable is $\ln y$ rather than y and the threshold `#` equals the minimum uncensored value of $\ln y$. The censored values of $\ln y$ must be set to a value equal to or less than the minimum uncensored value of $\ln y$.

In this model, interest lies in the prediction of expenditures in levels rather than logs. The issues are similar to those considered in section 3.4.8 for the lognormal model. Some algebra yields the censored mean

$$E(y|\mathbf{x}) = \exp\left(\mathbf{x}'\boldsymbol{\beta} + \frac{\sigma^2}{2}\right) \left\{ 1 - \Phi\left(\frac{\gamma - \mathbf{x}'\boldsymbol{\beta} - \sigma^2}{\sigma}\right) \right\} \quad (19.5)$$

The truncated mean $E(y|\mathbf{x}, y > 0)$ equals $E(y|\mathbf{x})/[1 - \Phi\{(\gamma - \mathbf{x}'\boldsymbol{\beta})/\sigma\}]$.

19.4.1 Data example

The illustrative application of the tobit model considered here uses the same data as in section [19.3](#). We remind the reader that in this sample of 3,328 observations, there are 526 (15.8%) zero values of `ambexp`. A detailed summary of $\ln(\text{ambexp})$, denoted by `lambexp`, follows.

```
. * Summary of log(expenditures) for positives only  
. summarize lambexp, detail
```

Natural logarithm of ambexp

	Percentiles	Smallest		
1%	3.091043	0		
5%	4.204693	.6931472		
10%	4.672829	.6931472	Obs	2,802
25%	5.616771	1.386294	Sum of wgt.	2,802
50%	6.65801		Mean	6.555066
		Largest	Std. dev.	1.41073
75%	7.556428	10.24952		
90%	8.285766	10.33916	Variance	1.990161
95%	8.704004	10.46207	Skewness	-.3421614
99%	9.43084	10.81898	Kurtosis	3.127747

The summary shows that, for positive values of `ambexp`, $\ln(\text{ambexp})$ is almost symmetrically distributed and has negligible nonnormal kurtosis. This is in stark contrast to `ambexp` even after conditioning on regressors; see section [19.3.1](#). We anticipate that the tobit model is better suited to modeling `lambexp` than `ambexp`.

19.4.2 Setting the censoring point for data in logs

It may be preferred at times to apply a transformation to the dependent variable to make it more suitable for a tobit application. In the present instance, we work with $\ln(\text{ambexp})$ as the dependent variable. This variable is originally set to missing if `ambexp = 0`, but to use the `tobit` command, we need to set it to a nonmissing value, the lower limit.

Here the smallest positive value of `ambexp` is 1, in which case $\ln(\text{ambexp})$ equals 0. Then Stata's default `ll` option or `ll(0)` option mistakenly treats this observation as censored rather than as zero, leading to shrinkage in the

sample size for noncensored observations. In our sample, one observation would be thus “lost”. To avoid this loss, we “tricked” Stata by setting all censored observations of $\ln y$ to an amount slightly smaller than the minimum noncensored value of $\ln y$, as follows:

```
. * "Tricking" Stata to handle log transformation and labeling variables
. generate y = ambexp
. generate dy = ambexp > 0
. generate lny = ln(y)                                // Zero values will become missing
(526 missing values generated)
. qui summarize lny
. scalar gamma = r(min)                            // This could be negative
. display "gamma = " gamma
gamma = 0
. replace lny = gamma - 0.0000001 if lny == .
(526 real changes made)
. tabulate y if y < 0.02                          // 0.02 is arbitrary small value


| y     | Freq. | Percent | Cum.   |
|-------|-------|---------|--------|
| 0     | 526   | 100.00  | 100.00 |
| Total | 526   | 100.00  |        |


. tabulate lny if lny < gamma + 0.02


| lny       | Freq. | Percent | Cum.   |
|-----------|-------|---------|--------|
| -1.00e-07 | 526   | 99.81   | 99.81  |
| 0         | 1     | 0.19    | 100.00 |
| Total     | 527   | 100.00  |        |


```

Note that the dependent variables have been relabeled. This makes the Stata code given later easier to adapt for other applications. In what follows, y is the `ambexp` variable, and $\ln y$ is the `lny` variable. Variables `y`, `dy`, and `lny` are already included in the dataset. The analysis below uses the scalar `gamma`, which equals -0.0000001 for these data.

```
. * Code below needs gamma which is the lower limit of lny (here 0) - 0.0000001
. scalar gamma = -0.0000001
```

19.4.3 Results

We first obtain the tobit MLE, where now log expenditures is the dependent variable.

```
. * Now do tobit on lny
. tobit lny $xlist, ll(gamma) vce(robust)

Refining starting values:
Grid node 0:    log likelihood = -7574.942
Fitting full model:
Iteration 0:    log pseudolikelihood = -7574.942
Iteration 1:    log pseudolikelihood = -7496.5907
Iteration 2:    log pseudolikelihood = -7494.2941
Iteration 3:    log pseudolikelihood = -7494.29
Iteration 4:    log pseudolikelihood = -7494.29

Tobit regression
Number of obs      = 3,328
Uncensored          = 2,802
Left-censored       = 526
Right-censored      = 0
F(6, 3322)          = 156.84
Prob > F            = 0.0000
Pseudo R2           = 0.0525

Log pseudolikelihood = -7494.29
```

	Robust					
lny	Coefficient	std. err.	t	P> t	[95% conf. interval]	
age	.3630699	.0457116	7.94	0.000	.2734441	.4526957
female	1.341809	.0991297	13.54	0.000	1.147447	1.53617
educ	.138446	.0201113	6.88	0.000	.0990142	.1778779
blhisp	-.8731611	.1174141	-7.44	0.000	-1.103372	-.6429499
totchr	1.161268	.053751	21.60	0.000	1.055879	1.266656
ins	.2612202	.0989029	2.64	0.008	.0673035	.4551369
_cons	.9237178	.3548763	2.60	0.009	.2279195	1.619516
var(e.lny)	7.735265	.2840619			7.197889	8.31276

All estimated coefficients are statistically significant at the 0.05 level and have the expected signs.

To assess the impact of using the censored regression framework instead of treating the zeros like observations from the same data-generating process as the positives, let us compare the results with those from the OLS regression of `lny` on the regressors.

```
. * OLS, not tobit
. regress lny $xlist, noheader vce(robust)
```

lny	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
age	.3247317	.0388171	8.37	0.000	.2486239	.4008396
female	1.144695	.0831535	13.77	0.000	.9816578	1.307733
educ	.114108	.0165032	6.91	0.000	.0817505	.1464654
blhisp	-.7341754	.0974335	-7.54	0.000	-.9252112	-.5431397
totchr	1.059395	.0462565	22.90	0.000	.968701	1.150089
ins	.2078343	.0840858	2.47	0.013	.0429692	.3726995
_cons	1.728764	.2865494	6.03	0.000	1.166933	2.290596

All the OLS slope coefficients are in absolute terms smaller than those for the ML tobit, the reduction being 10–15%, but the OLS intercept is larger. The impact of censoring (zeros) on the OLS results depends on the proportion of censored observations, which in our case is around 15%.

19.4.4 Two-limit tobit

In less than 1.5% of the sample (48 observations), ambexp exceeds \$10,000. Suppose that we want to exclude these high values that contribute to the nonnormal kurtosis. Or suppose that the data above an upper-cutoff point are reported as falling in an interval. Choosing \$10,000 as the upper-censoring point, we fit a two-limit tobit version of the tobit model. We see that the impact of dropping the 48 observations is relatively small. This is not too surprising because a small proportion of the sample is right-censored.

```

. * Now do two-limit tobit
. scalar upper = log(10000)
. display upper
9.2103404
. tobit lny $xlist, ll(gamma) ul(9.2103404) vce(robust)

Refining starting values:
Grid node 0:  log likelihood = -7544.0716

Fitting full model:
Iteration 0:  log pseudolikelihood = -7544.0716
Iteration 1:  log pseudolikelihood = -7453.8109
Iteration 2:  log pseudolikelihood = -7451.7643
Iteration 3:  log pseudolikelihood = -7451.7623
Iteration 4:  log pseudolikelihood = -7451.7623

Tobit regression
Number of obs      = 3,328
Uncensored          = 2,754
Left-censored       = 526
Right-censored      = 48
F(6, 3322)          = 149.00
Prob > F            = 0.0000
Pseudo R2           = 0.0534

Log pseudolikelihood = -7451.7623

```

lny	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
age	.3711061	.0463719	8.00	0.000	.2801858	.4620264
female	1.348768	.1003897	13.44	0.000	1.151936	1.5456
educ	.1402643	.0203227	6.90	0.000	.1004181	.1801105
blhisp	-.8759505	.1187037	-7.38	0.000	-1.10869	-.6432108
totchr	1.20494	.0571746	21.07	0.000	1.092839	1.317041
ins	.2466838	.1001165	2.46	0.014	.0503875	.4429802
_cons	.8638458	.3591707	2.41	0.016	.1596275	1.568064
var(e.lny)	7.909052	.2943484			7.352483	8.507753

19.4.5 Model diagnostics

To test the validity of the key tobit assumptions of normality and homoskedasticity, we need to apply some diagnostic checks. For the ordinary linear regression model, the `sktest` and `estat hettest` commands are available to test for normality and homoskedasticity. These tests are based on the OLS residuals. These postestimation tests are invalid for censored data because the fitted values and residuals from a censored model do not share the properties of their ordinary regression counterparts. Instead,

generalized residuals for censored regression, as discussed in [Cameron and Trivedi \(2005\)](#), 630) and in [Verbeek \(2017\)](#), 238–240), provide the key component for generating test statistics for testing the null hypotheses of homoskedasticity and normality.

In linear regression, tests of homoskedasticity typically use squared residuals, and tests of normality use residuals raised to a power of 3 or 4. The first step is then to construct analogous quantities for the censored regression.

For uncensored observations, we use $\hat{\varepsilon}_i = (y_i - \mathbf{x}'_i \hat{\beta})/\hat{\sigma}$ raised to the relevant powers, where y_i is a generic notation for the dependent variable, which here is $\ln(\text{ambexp})$.

For observations truncated from above, with $\ln y_i^* < \gamma$ and $d_i = 0$, we use the truncated moment of the normal error listed in table [19.1](#), evaluated at $\hat{\beta}$ and $\hat{\sigma}$.

Table 19.1. Moments of normal errors truncated from above

Moments	Expression
$E(\varepsilon_i d_i = 0)$	$-\lambda_i$
$E(\varepsilon_i^2 d_i = 0)$	$-z_i \lambda_i$
$E(\varepsilon_i^3 d_i = 0)$	$-(2 + z_i^2) \lambda_i$
$E(\varepsilon_i^4 d_i = 0)$	$-(3z_i + z_i^3) \lambda_i$

notes: $\lambda_i = \phi(z_i)/\Phi(z_i)$ and $z_i = (\gamma - \mathbf{x}'_i \beta)/\sigma$.

The components $\phi(\cdot)$ and $\Phi(\cdot)$ given in table [19.1](#) can be evaluated by using Stata's `normalden()` and `normal()` functions. Given these and predicted values from the ML regression, the four “generalized” components given in the table can be readily computed.

19.4.6 Tests of normality and homoskedasticity

Lagrange multiplier, or score, tests of heteroskedasticity and nonnormality are appealing because they require estimation only of the models under the hypothesis of normality and homoskedasticity. The test statistics are quadratic forms that can be calculated in several different ways. One way is by using an auxiliary regression; see section 11.5.3 and [Cameron and Trivedi \(2005, chap. 8\)](#).

Conditional moment tests can also be performed by using a similar approach; see section 11.9.1, [Newey \(1985\)](#), and [Pagan and Vella \(1989\)](#). Such regression-based tests have been developed with generalized residuals. Although they are not currently available as a part of the official Stata package, they can be constructed from Stata output, as illustrated below. The key component of the auxiliary regression is the uncentered R^2 , denoted by R_u^2 , from the auxiliary regression of 1 on generated regressors that are themselves functions of generalized residuals. The specific regressors depend upon the alternative to the null.

We implement two conditional moment tests below. The first is a test of normality based on testing whether the uncensored error ε_i has third moment zero and fourth moment $3\sigma^2$, as is the case for the normal distribution. Thus, we test $E\{(y_i^* - \mathbf{x}'_i \boldsymbol{\beta})^3\} = 0$ and $E\{(y_i^* - \mathbf{x}'_i \boldsymbol{\beta})^4 - 3\sigma^2\} = 0$. The heteroskedasticity test is a test of $E\{\mathbf{w}_i(y_i^* - \mathbf{x}'_i \boldsymbol{\beta})^2\} = \mathbf{0}$, where \mathbf{w}_i are variables that σ^2 may vary with if there is heteroskedasticity.

Generalized residuals and scores

To implement the test, we first need to compute the inverse of Mills's ratio, λ_i , because from table [19.1](#), the generalized residuals depend in part on λ_i .

```
. * Compute Mills's ratio
. qui tobit lny $xlist, ll(gamma) vce(robust)
. predict xb, xb                                // xb is estimate of x'b
. matrix btobit = e(b)
. scalar sigma = sqrt(btobit[1,e(df_m)+2])    // sigma is estimate of sigma
. generate threshold = (gamma-xb)/sigma        // gamma: lower censoring point
. generate lambda = normalden(threshold)/normal(threshold)
```

Next, we calculate generalized residuals and functions of them. For example, `gres3`, the sample analogue of $\{(y_i^* - \mathbf{x}'_i \boldsymbol{\beta})/\sigma\}^3$, equals $\{(y_i - \mathbf{x}'_i \hat{\boldsymbol{\beta}})/\hat{\sigma}\}^3$ for an uncensored observation and equals $-(2 + \hat{z}_i^2)\hat{\lambda}_i$ for a censored observation, where z_i and λ_i are defined in the table. The generalized residuals `gres1` and `gres2` can be shown to be the contributions to the score for, respectively, the intercept β_1 and σ , so they must sum to zero over the sample. The generalized residuals `gres3` and `gres4` satisfy the same zero-mean property only if the model is correctly specified.

The generalized residuals are computed as follows:

```
. * Generalized residuals: First create gres1, which has mean zero, by the f.o.c.
. qui generate uifdyeq1 = (lny - xb)/sigma if dy == 1
. qui generate double gres1 = uifdyeq1
. qui replace gres1 = -lambda if dy == 0
. summarize gres1
```

Variable	Obs	Mean	Std. dev.	Min	Max
gres1	3,328	-7.03e-10	.9877495	-3.129662	2.245604

The zero-mean property of `gres1` is thus verified. The remaining three variables are computed next.

```
. * Generalized residuals: Next create gres2, gres3, and gres4
. qui generate double gres2 = uifdyeq1^2 - 1
. qui replace gres2 = -threshold*lambda if dy == 0
. qui generate double gres3 = uifdyeq1^3
. qui replace gres3 = -(2 + threshold^2)*lambda if dy == 0
. qui generate double gres4 = uifdyeq1^4 - 3
. qui replace gres4 = -(3*threshold + threshold^3)*lambda if dy == 0
```

Test of normality

To apply the auxiliary regression to implement the test for normality, we need to calculate the likelihood scores that are included as additional regressors; see section 11.5.3. The components of the scores with respect to $\boldsymbol{\beta}$ are $\hat{\lambda}_i$ times the relevant component of \mathbf{x} , that is, $\hat{\lambda}_i \mathbf{x}_i$. These can be computed by using the `foreach` command:

```

. * Generate the scores to use in the Lagrange multiplier test of normality
. foreach var in $xlist {
2.     generate score`var' = gres1*`var'
3. }
. global scores score* gres1 gres2

```

Recall that `gres1` is the score with respect to the intercept β_1 and `gres2` is the score with respect to the intercept σ . So the global `scores` contains all the scores with respect to β and σ^2 for the tobit model.

To execute the regression-based test of normality, a test based on the third and fourth moments of ε_i , we regress one on `gres3` and `gres4`, along with `scores`, and compute the NR^2 statistic. Because two moment conditions are being tested, the test statistic is $\chi^2(2)$ under the null hypothesis; the remaining variables in the regression are scores that need to be included for the test based on the auxiliary regression to be valid.

```

. * Test of normality in tobit regression uses NR^2 from uncentered regression
. generate one = 1
. qui regress one gres3 gres4 $scores, noconstant
. display "N R^2 = " e(N)*e(r2) " with p-value = " chi2tail(2,e(N)*e(r2))
N R^2 = 1832.128 with p-value = 0

```

The outcome of the test is a very strong rejection of the normality hypothesis, even though the expenditure variable was transformed to logarithms.

The properties of the conditional moment test as implemented here have been investigated by [Skeels and Vella \(1999\)](#), who found that using the asymptotic distribution of this test produces severe size distortions, even in moderately large samples. This is an important limitation of the test. [Drukker \(2002\)](#) developed a parametric bootstrap to correct the size distortion by using bootstrap critical values. His Monte Carlo results show that the test based on bootstrap critical values has reasonable power for samples larger than 500.

The community-contributed `tobcm` command ([Drukker 2002](#)) implements this better variant of the test. The command works only after `tobit` and with the left-censoring point 0 and no right-censoring. To compare the above outcome of the normality test with that from the

improved bootstrap version, the interested reader can perform the `tobcm` command quite easily.

Test of homoskedasticity

For testing homoskedasticity, the alternative hypothesis is that the variance of ε_i is of the form $\sigma^2 \exp(\mathbf{w}'_i \boldsymbol{\alpha})$, and we test whether

$E\{\mathbf{w}_i(y_i^* - \mathbf{x}'_i \boldsymbol{\beta})^2\} = 0$. This leads to an auxiliary regression of 1 on $\mathbf{w}_i \times \text{gres2}_i$; often, \mathbf{w} is specified to be the same as \mathbf{x} , though necessarily excludes the intercept. If $\dim(\mathbf{w}_i) = K$, then $\text{NR}_u^2 \sim \chi^2(K)$ under the null hypothesis.

The following test sets $\mathbf{w} = \mathbf{x}$, aside from the intercept, so 6 moment conditions are tested, and inference is based on the $\chi^2(6)$ distribution; the remaining variables in the regression are scores that need to be included for the test based on the auxiliary regression to be valid.

```
. * Test of homoskedasticity in tobit regression with w=x (aside from intercept)
. foreach var in $xlist {
    2. generate gres2by`var' = gres2*`var'
    3. }
. qui regress one gres2by* $scores, noconstant
. display "N R^2 = " e(N)*e(r2) " with p-value = " chi2tail(6,e(N)*e(r2))
N R^2 = 596.73953 with p-value = 1.18e-125
```

There is strong rejection of the null hypothesis of homoskedasticity against the alternative that the variance is of the form specified. If an investigator wants to specify different components of \mathbf{w} , then the required modifications to the above commands are trivial.

19.4.7 Next step?

Despite the apparently satisfactory estimation results for the tobit model, the diagnostic tests reveal weaknesses. The failure of normality and homoskedasticity assumptions has serious consequences for censored-data regression that do not arise in the case of linear regression. A natural question that arises concerns the direction in which additional modeling effort might be directed to arrive at a more general model.

Two approaches to such generalization will be considered. The two-part model, given in the next section, specifies one model for the censoring mechanism and a second distinct model for the outcome conditional on the outcome being observed. The sample-selection model, presented in the subsequent section, instead specifies a joint distribution for the censoring mechanism and outcome and then finds the implied distribution conditional on the outcome observed.

19.5 Two-part model in logs

The tobit regression makes a strong assumption that the same probability mechanism generates both censored and uncensored observations, where throughout we presume censoring at $y = 0$. It is more flexible to allow for the possibility that the zero and positive values are generated by different mechanisms. Many applications have shown that an alternative model, the two-part model or the hurdle model, can provide a better fit by relaxing the tobit model assumptions.

This model is the natural next step in our modeling strategy. Again, we apply it to a model in logs rather than levels.

19.5.1 Model structure

The first part of the model is a binary outcome equation that models $\Pr(\text{ambexp} > 0)$, using any of the binary outcome models considered in chapter 17 (usually probit). The second part uses linear regression to model $E(\ln \text{ambexp} | \text{ambexp} > 0)$. The two parts are assumed to be independent and are usually estimated separately.

Let y denote ambexp . Define a binary indicator, d , of positive expenditure such that $d = 1$ if $y > 0$ and $d = 0$ if $y = 0$. When $y = 0$, we model $\Pr(d = 0)$. For those with $y > 0$, let $f(y|d = 1)$ be the conditional density of y . The two-part model for y is then given by

$$f(y|\mathbf{x}) = \begin{cases} \Pr(d = 0|\mathbf{x}) & \text{if } y = 0 \\ \Pr(d = 1|\mathbf{x})f(y|d = 1, \mathbf{x}) & \text{if } y > 0 \end{cases} \quad (19.6)$$

The same regressors often appear in both parts of the model, but this can and should be relaxed if there are obvious exclusion restrictions.

The probit or the logit is an obvious choice for the first part. If a probit model is used, then $\Pr(d = 1|\mathbf{x}) = \Phi(\mathbf{x}'_1 \boldsymbol{\beta}_1)$. If a lognormal model for

$y|y > 0$ is given, then $(\ln y|d = 1, \mathbf{x}) \sim N(\mathbf{x}'_2 \boldsymbol{\beta}_2, \sigma_2^2)$. Combining these, we have for the model in logs

$$E(y|\mathbf{x}_1, \mathbf{x}_2) = \Phi(\mathbf{x}'_1 \boldsymbol{\beta}_1) \exp(\mathbf{x}'_2 \boldsymbol{\beta}_2 + \sigma_2^2/2)$$

where the second term uses the result that if $\ln y \sim N(\mu, \sigma^2)$, then $E(y) = \exp(\mu + \sigma^2/2)$.

ML estimation of (19.6) is straightforward because it separates the estimation of a discrete choice model using all observations and the estimation of the parameters of the density $f(y|d = 1, \mathbf{x})$ using only the observations with $y > 0$.

19.5.2 Part 1 specification

In the example considered here, $\mathbf{x}_1 = \mathbf{x}_2$, but there is no reason why this should always be so. It is an advantage of the two-part model that it provides the flexibility to have different regressors in the two parts. In this example, the first part is modeled through a probit regression, and again one has the flexibility to change this to logit or complementary log–log regression. Comparing the results from the tobit, two-part, and selection models is a little easier if we use the probit form.

```

. * Part 1 of the two-part model
. probit dy $xlist, nolog vce(robust)

Probit regression
Number of obs = 3,328
Wald chi2(6) = 318.08
Prob > chi2 = 0.0000
Pseudo R2 = 0.1754

Log pseudolikelihood = -1197.6644

```

dy	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
age	.097315	.0272976	3.56	0.000	.0438127	.1508173
female	.6442089	.0610491	10.55	0.000	.524555	.7638629
educ	.0701674	.0108653	6.46	0.000	.0488718	.0914631
blhisp	-.3744867	.0610172	-6.14	0.000	-.4940781	-.2548952
totchr	.7935208	.0739854	10.73	0.000	.6485121	.9385294
ins	.1812415	.0611857	2.96	0.003	.0613198	.3011632
_cons	-.7177087	.1860721	-3.86	0.000	-1.082403	-.3530141

```
. scalar llprobit = e(l1)
```

The probit regression indicates that all covariates are statistically significant determinants of the probability of positive expenditure. The standard ME calculations can be done for the first part, as illustrated in chapter [17](#).

19.5.3 Part 2 of the two-part model

The second part is a linear regression of `lny`, here `ln(ambexp)`, on the regressors in the global macro `xlist`.

```
. * Part 2 of the two-part model
. regress lny $xlist if dy==1, vce(robust)

Linear regression
Number of obs      =      2,802
F(6, 2795)        =     134.62
Prob > F          =     0.0000
R-squared          =     0.1918
Root MSE           =     1.2696
```

lny	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
age	.2172327	.0220962	9.83	0.000	.1739062	.2605592
female	.3793756	.0489726	7.75	0.000	.2833494	.4754018
educ	.0222388	.0096629	2.30	0.021	.0032917	.0411859
blhisp	-.2385321	.0560156	-4.26	0.000	-.3483682	-.1286961
totchr	.5618171	.028185	19.93	0.000	.5065516	.6170826
ins	-.020827	.0488003	-0.43	0.670	-.1165153	.0748614
_cons	4.907825	.1717797	28.57	0.000	4.570997	5.244653

```
. scalar lllognormal = e(l1)
. predict rlambexp, residuals
```

The coefficients of regressors in the second part have the same sign as those in the first part, aside from the `ins` variable, which is highly statistically insignificant in the second part.

Given the assumption that the two parts are independent, the joint likelihood for the two parts is the sum of two log likelihoods, that is, $-5,838.8$. The computation is shown below.

```
. * Create two-part model log likelihood
. scalar lltwopart = llprobit + lllognormal // Two-part model log likelihood
. display "lltwopart = " lltwopart
lltwopart = -5838.8218
```

By comparison, the log likelihood for the tobit model is $-7,494.29$. The two-part model fits the data considerably better, even if Akaike's information criterion or Bayesian information criterion is used to penalize the two-part model for its additional parameters.

Does the two-part model eliminate the twin problems of heteroskedasticity and nonnormality? This is easily checked using the `estat hettest` and `sktest` commands.

```
. * hettest and sktest commands require default standard errors
. qui regress lny $xlist if dy==1
. hettest
```

Breusch-Pagan/Cook-Weisberg test for heteroskedasticity
Assumption: Normal error terms
Variable: Fitted values of lny

H0: Constant variance

```
chi2(1) = 19.25
Prob > chi2 = 0.0000
```

```
. sktest rlambexp
```

Skewness and kurtosis tests for normality

Variable	Obs	Pr(skewness)	Pr(kurtosis)	Joint test	
				Adj chi2(2)	Prob>chi2
rlambexp	3,328	0.0000	0.0592	368.86	0.0000

The tests on log expenditures for those with positive expenditures unambiguously reject the homoskedasticity and normality hypotheses. However, unlike the tobit model, neither condition is necessary for consistency of the estimator because the key assumption needed is that $E(\ln y|d = 1, \mathbf{x})$ is linear in \mathbf{x} . On the other hand, it is known that the OLS estimate of the residual variance will be biased in the presence of heteroskedasticity. This deficiency will extend to those predictors of y that involve the residual variance. This point is pursued further in an end-of-chapter exercise.

From the viewpoint of interpretation, the two-part model is flexible and attractive because it allows different covariates to have a different impact on the two parts of the model. For example, it allows a variable to make its impact entirely by changing the probability of a positive outcome, with no impact on the size of the outcome conditional on it being positive. In our example, the coefficient of `ins` in the conditional regression has a small and statistically insignificant coefficient but has a positive and significant coefficient in the probit equation.

19.6 Selection models

The two-part model attains some of its flexibility and computational simplicity by assuming that the two parts—the decision to spend and the amount spent—are independent. Then selection is only on observables.

This is a potential restriction on the model. If it is conceivable that, after one controls for regressors, those with positive expenditure levels are not randomly selected from the population, then the results of the second-stage regression suffer from selection bias. In that case, selection is on both observables and unobservables.

The selection model used in this section considers the possibility of such bias by allowing for possible dependence across the two parts of the model. This new model is an example of a bivariate sample-selection model. It is known variously as the type-2 tobit model, the heckit model, and incidental truncation with a probit selection equation.

There are several different ways that selection on unobservables may arise leading to different selection models. The type-2 tobit model is the most commonly used model and is used as one method to control for panel attrition in section [19.11](#). Other selection models include the endogenous switching regimes model given in section [19.9.3](#).

A censored sample is one where the dependent variable is incompletely observed. Instead of observing (y_i, \mathbf{x}_i) , we observe $\{h(y_i), \mathbf{x}_i\}$, where examples of $h(\cdot)$ include left-censoring at 0 [so $h(\cdot) = \min(0, y_i)$], right-censoring, interval-censoring, and observing only a binary outcome. The `tobit`, `intreg`, and `probit` commands cover these cases if y is conditionally normally distributed.

Sample selection is instead the case where data on some observations are missing. A simple example is truncation at 0 where we observe (y_i^*, \mathbf{x}_i) only if $y_i^* > 0$. Another simple example is where data on \mathbf{x}_i , or a subcomponent of \mathbf{x}_i , are missing for some observations. A more complicated example is the selection model of this section where we may observe y_i if a related variable crosses a threshold.

The application in this section uses expenditures in logs. The same method can be applied without modification to expenditures in levels.

19.6.1 Model structure and assumptions

Throughout this section, an asterisk will denote a latent variable. Let y_2^* denote the outcome of interest, here expenditure or log expenditure. In the standard tobit model, this outcome is observed if $y_2^* > 0$. A more general model introduces a second latent variable, y_1^* , and the outcome y_2^* is observed if $y_1^* > 0$. In the present case, y_1^* determines whether an individual has any ambulatory expenditure, y_2^* determines the level or logarithm of expenditure, and $y_1^* \neq y_2^*$.

The two-equation model comprises a selection equation for y_1 , where

$$y_1 = \begin{cases} 1 & \text{if } y_1^* > 0 \\ 0 & \text{if } y_1^* \leq 0 \end{cases}$$

and a resultant outcome equation for y_2 , where

$$y_2 = \begin{cases} y_2^* & \text{if } y_1^* > 0 \\ - & \text{if } y_1^* \leq 0 \end{cases}$$

Here y_2 is observed only when $y_1^* > 0$, possibly taking a negative value, whereas y_2 need not take on any meaningful value when $y_1^* \leq 0$. The classic version of the model is linear with additive errors, so

$$\begin{aligned} y_1^* &= \mathbf{x}'_1 \boldsymbol{\beta}_1 + \varepsilon_1 \\ y_2^* &= \mathbf{x}'_2 \boldsymbol{\beta}_2 + \varepsilon_2 \end{aligned}$$

with ε_1 and ε_2 possibly correlated. The tobit model is a special case where $y_1^* = y_2^*$.

It is assumed that the correlated errors are jointly normally distributed and homoskedastic; that is,

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix} \sim N \left[\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix} \right]$$

where the normalization $\sigma_1^2 = 1$ is used because only the sign of y_1^* is observed. Estimation by ML is straightforward.

The likelihood function for this model is

$$L = \prod_{i=1}^N \{\Pr(y_{1i}^* \leq 0)\}^{1-y_{1i}} \{f(y_{2i} | y_{1i}^* > 0) \times \Pr(y_{1i}^* > 0)\}^{y_{1i}}$$

where the first term is the contribution when $y_{1i}^* \leq 0$, because then $y_{1i} = 0$, and the second term is the contribution when $y_{1i}^* > 0$. This likelihood function can be specialized to models other than the linear model considered here. In the case of linear models with jointly normal errors, the bivariate density, $f^*(y_1^*, y_2^*)$, is normal, and hence the conditional density in the second term is univariate normal.

The essential structure of the model and the ML estimation procedure are not affected by the decision to model positive expenditure on the log (rather than the linear) scale, although this does affect the conditional prediction of the level of expenditure. This step is taken here even though tests implemented in the previous two sections show that the normality and homoskedasticity assumptions are both questionable.

19.6.2 ML estimation of the sample-selection model

ML estimation of the bivariate sample-selection model with the `heckman` command is straightforward. The basic syntax for this command is

```
heckman depvar [indepvars] [if] [in] [weight],  
    select([depvar_s =] varlist_s [, noconstant offset(varname_o)]) [options]
```

where `select()` is the option for specifying the selection equation. One needs to specify variable lists for both the selection equation and for the outcome equation. In many cases, the investigator might use the same set of regressors in both equations. When this is done, it is often referred to as the case in which model identification is based solely upon the nonlinearity in the functional form. Because the selection equation is nonlinear, it potentially allows the higher powers of regressors to affect the selection variable. In the linear outcome equation, of course, the higher powers do not appear. Therefore, the nonlinearity of the selection regression automatically generates exclusion restrictions. That is, it allows for an independent source of variation in the probability of a positive outcome, hence the term “identification through nonlinear functional form”.

The specification of the selection equation involves delicate identification issues. For example, if the nonlinearity implied by the probit model is slight, then the identification will be fragile. Thus, it is common in applied work to look for exclusion restrictions. The investigator seeks variables that can generate nontrivial variation in the selection variable but do not affect the outcome variable directly. This is exactly the same argument as was encountered in earlier chapters in the context of instrumental variables (IV). A valid exclusion restriction arises if a suitable instrument is available, and this may vary from case to case. We will illustrate the practical importance of these ideas in the examples that follow.

An alternative to the `heckman` command is available within the family of extended regression model (ERM) commands presented in section [23.7](#). This alternative command, `eregress`, uses the option `select()` and has syntax similar to the `heckman` command as in, for example, `eregress lny x, select(dy = x z) nolog`, where `z` is an exogenous variable in the selection equation but excluded from the outcome equation.

Sample selection extends beyond continuous outcome models. For example, binary outcomes may be subject to sample selection also; that is, the probability of observing a 0/1 outcome may be subject to sampling bias. Given that the probit model is based on the normal distribution, Heckman’s

approach for handling sample selection in the standard normal regression can be extended also to the probit regression. Stata's command `heckprobit` will handle such a case; its syntax is similar to the `heckman` command. Alternatively, the equivalent `eprobit` command can be used. A further extension to ordered probit models with sample selection is also feasible using the `heckoprobit` command or the equivalent `eoprobit` command, which is also a member of the ERM suite of commands based on the recursive structure assumption. The `heckpoisson` command accommodates selection in the Poisson model.

19.6.3 Estimation without exclusion restrictions

We first estimate the parameters of the selection model without exclusion restrictions.

```

. * Heckman MLE without exclusion restrictions
. heckman lny $xlist, select(dy = $xlist) nolog vce(robust)

Heckman selection model
(regression model with sample selection)
Number of obs      =      3,328
Selected          =      2,802
Nonselected       =       526
Wald chi2(6)      =     517.92
Prob > chi2       =    0.0000
Log pseudolikelihood = -5838.397

```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
lny	age	.2122921	.0225168	9.43	0.000 .16816 .2564242
	female	.349728	.053152	6.58	0.000 .2455519 .453904
	educ	.0188724	.0099991	1.89	0.059 -.0007256 .0384704
	blhisp	-.2196042	.057581	-3.81	0.000 -.332461 -.1067474
	totchr	.5409537	.029986	18.04	0.000 .4821822 .5997252
	ins	-.0295368	.0491937	-0.60	0.548 -.1259547 .0668811
	_cons	5.037418	.1976607	25.49	0.000 4.65001 5.424826
dy	age	.0984482	.02728	3.61	0.000 .0449804 .151916
	female	.6436686	.0610712	10.54	0.000 .5239713 .7633659
	educ	.0702483	.0108621	6.47	0.000 .0489589 .0915377
	blhisp	-.3726284	.0610389	-6.10	0.000 -.4922625 -.2529944
	totchr	.7946708	.0738284	10.76	0.000 .6499697 .9393718
	ins	.1821233	.0611301	2.98	0.003 .0623106 .301936
	_cons	-.7244413	.1862954	-3.89	0.000 -1.089574 -.359309
/athrho		-.124847	.0949188	-1.32	0.188 -.3108845 .0611905
	/lnsigma	.2395983	.0152022	15.76	0.000 .2098026 .2693941
	rho	-.1242024	.0934546		-.3012415 .0611142
	sigma	1.270739	.019318		1.233435 1.309171
	lambda	-.1578287	.1190885		-.3912379 .0755805

Wald test of indep. eqns. (rho = 0): chi2(1) = 1.73 Prob > chi2 = 0.1884

The log likelihood for this model is very slightly higher than that for the two-part model: $-5,838.4$ compared with $-5,838.8$ (see section [19.5.3](#)). Consistent with this small difference is the finding that $\hat{\rho} = -0.124$ with the 95% confidence interval $[-0.301, 0.061]$. The Wald test has a p -value of 0.188.

Thus, the estimated correlation between the errors is not significantly different from zero, and the hypothesis that the two parts are independent cannot be rejected. This is an important result. For some types of censored

data, notably, hours of work, it is necessary to control for selection, and it can make a big difference. For other types of data, such as medical expenditure data, there is often less of a selection problem, and a two-part model may be sufficient.

The foregoing conclusion should be treated with caution because the model is based on a bivariate normality assumption that is itself suspect. The two-step estimation, considered next, relies on a univariate normality assumption and is expected to be relatively more robust.

19.6.4 Two-step estimation

The MLE in the sample-selection model assumes joint normality of the errors ε_1 and ε_2 . This assumption can be relaxed. Instead, we need assume only that

$$\begin{aligned}\varepsilon_1 &\sim N(0, 1) \\ \varepsilon_2 &= \sigma_{12}\varepsilon_1 + \eta\end{aligned}$$

where η is an independent error. The outcome equation error is a multiple of the selection equation error, which is standard normal distributed, plus noise.

It can be shown that these assumptions imply that

$$E(y_2|\mathbf{x}, y_1^* > 0) = \mathbf{x}'_2 \boldsymbol{\beta}_2 + \sigma_{12}\lambda(\mathbf{x}'_1 \boldsymbol{\beta}_1) \quad (19.7)$$

where $\lambda(\cdot) = \phi(\cdot)/\Phi(\cdot)$. The motivation is that $y_2^* = \mathbf{x}'_2 \boldsymbol{\beta}_2 + \varepsilon_2$ so that given selection $E(y_2|\mathbf{x}, y_1^* > 0) = \mathbf{x}'_2 \boldsymbol{\beta}_2 + E(\varepsilon_2|y_1^* > 0)$. Given the above assumptions,

$E(\varepsilon_2|y_1^* > 0) = E(\varepsilon_2|\varepsilon_1 > -\mathbf{x}'_1 \boldsymbol{\beta}_1) = \sigma_{12}E(\varepsilon_1|\varepsilon_1 > -\mathbf{x}'_1 \boldsymbol{\beta}_1) = \sigma_{12}\lambda(\mathbf{x}'_1 \boldsymbol{\beta}_1)$, where the last result uses the formula for the left-truncated mean of the standard normal.

The second term in (19.7) can be estimated by $\lambda(\mathbf{x}'_1 \widehat{\boldsymbol{\beta}}_1)$, where $\widehat{\boldsymbol{\beta}}_1$ is obtained by probit regression of y_1 on \mathbf{x}_1 . The OLS regression of y_2 on \mathbf{x}_2 and

the generated regressor, $\lambda(\mathbf{x}_1' \hat{\boldsymbol{\beta}}_1)$, called the inverse of Mills's ratio or the nonselection hazard, yields a semiparametric estimate of $(\boldsymbol{\beta}_2, \sigma_{12})$. The calculation of the standard errors, however, is complicated by the presence in the regression of the generated regressor, $\lambda(\mathbf{x}_1' \hat{\boldsymbol{\beta}}_1)$.

The addition of the `twostep` option to `heckman` yields the two-step estimator.

```
. * Heckman two-step without exclusion restrictions
. heckman lny $xlist, select(dy = $xlist) twostep

Heckman selection model -- two-step estimates      Number of obs      =      3,328
(regression model with sample selection)          Selected      =      2,802
                                                 Nonselected    =       526
                                                 Wald chi2(6)    =     189.46
                                                 Prob > chi2    =     0.0000
```

	Coefficient	Std. err.	z	P> z	[95% conf. interval]
lny					
age	.202124	.0242974	8.32	0.000	.1545019 .2497462
female	.2891575	.073694	3.92	0.000	.1447199 .4335951
educ	.0119928	.0116839	1.03	0.305	-.0109072 .0348928
blhisp	-.1810582	.0658522	-2.75	0.006	-.3101261 -.0519904
totchr	.4983315	.0494699	10.07	0.000	.4013724 .5952907
ins	-.0474019	.0531541	-0.89	0.373	-.151582 .0567782
_cons	5.302572	.2941363	18.03	0.000	4.726076 5.879069
dy					
age	.097315	.0270155	3.60	0.000	.0443656 .1502645
female	.6442089	.0601499	10.71	0.000	.5263172 .7621006
educ	.0701674	.0113435	6.19	0.000	.0479345 .0924003
blhisp	-.3744867	.0617541	-6.06	0.000	-.4955224 -.2534509
totchr	.7935208	.0711156	11.16	0.000	.6541367 .9329048
ins	.1812415	.0625916	2.90	0.004	.0585642 .3039187
_cons	-.7177087	.1924667	-3.73	0.000	-1.094937 -.3404809
/mills					
lambda	-.4801696	.2906565	-1.65	0.099	-1.049846 .0895067
rho	-0.37130				
sigma	1.2932083				

The reported standard errors for the regression coefficients, $\hat{\boldsymbol{\beta}}_2$, control for the estimation error of $\lambda(\mathbf{x}_1' \hat{\boldsymbol{\beta}}_1)$; see [R] **heckman**. These standard errors are in general larger than those from the ML estimation. Although no standard

error is provided for `rho=lambda/sigma`, the hypothesis of independence of ε_1 and ε_2 can be tested directly by using the coefficient of `lambda`, because from (19.7), this is the error covariance σ_{12} . The coefficient of `lambda` has a larger z statistic, -1.65 , than in the ML case, and it is significantly different from 0 at any p -value higher than 0.099. Thus, the two-step estimator produces somewhat stronger evidence of selection than does the ML estimator.

The `heckman` command standard errors are based on analytical results and are not robust. Robust standard errors can be obtained by a bootstrap, as done for this example in section 12.4.5.

Alternatively, we can stack the moment conditions for the two steps and estimate by just-identified generalized method of moments (GMM). The estimating equations for the first-step ML probit and the second step OLS estimation are, respectively,

$$\begin{aligned} \sum_{i=1}^N \frac{y_{1i} - \Phi(\mathbf{x}'_{ii}\boldsymbol{\beta}_1)}{\Phi(\mathbf{x}'_{1i}\boldsymbol{\beta}_1)\{1 - \Phi(\mathbf{x}'_{1i}\boldsymbol{\beta}_1)\}} \phi(\mathbf{x}'_{1i}\boldsymbol{\beta}_1) \mathbf{x}_{1i} &= \mathbf{0} \\ \sum_{i=1}^N y_{1i} \{y_{2i} - \mathbf{x}'_{2i}\boldsymbol{\beta}_2 - \sigma_{12}\lambda(\mathbf{x}'_{1i}\boldsymbol{\beta}_1)\} \begin{bmatrix} \mathbf{x}'_{2i} \\ \lambda(\mathbf{x}'_{1i}\boldsymbol{\beta}_1) \end{bmatrix} &= \mathbf{0} \end{aligned}$$

This model can be fit using the `gmm` command by a method similar to that in section 13.3.11 for the control function estimator in the linear model.

19.6.5 Estimation with exclusion restrictions

The standard errors of the two-step estimator are larger than those of the ML estimator in part because the variable $\lambda(\mathbf{x}'_1\hat{\boldsymbol{\beta}}_1)$ can be collinear with the other regressors in the outcome equation (\mathbf{x}_2). This is highly likely if $\mathbf{x}_1 = \mathbf{x}_2$, as would be the case when there are no exclusion restrictions. Having exclusion restrictions, so that $\mathbf{x}_1 \neq \mathbf{x}_2$, may reduce the collinearity problem, especially in small samples.

For more robust identification, it is usually recommended, as has been explained above, that exclusion restrictions be imposed. This requires that

the selection equation has an exogenous variable that is excluded from the outcome equation. Moreover, the excluded variable should have a substantial (nontrivial) impact on the probability of selection. Because it is often hard to come up with an excluded variable that does not directly affect the outcome and does affect the selection, the investigator should have strong justification for imposing the exclusion restriction.

We repeat the ML computation of the Heckman model with an additional regressor, `income`, in the selection equation.

```

. * Heckman MLE with exclusion restriction
. heckman lny $xlist, select(dy = $xlist income) nolog vce(robust)
Heckman selection model
(regression model with sample selection)
Number of obs      =      3,328
Selected          =      2,802
Nonselected       =       526
Wald chi2(6)      =     484.20
Prob > chi2       =    0.0000
Log pseudolikelihood = -5836.219

```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
lny	age	.2119749	.0225429	9.40	0.000 .2561583
	female	.3481441	.0541676	6.43	0.000 .4543105
	educ	.018716	.0099997	1.87	0.061 .038315
	blhisp	-.2185714	.0578622	-3.78	0.000 -.1051637
	totchr	.53992	.030624	17.63	0.000 .5999419
	ins	-.0299871	.0492054	-0.61	0.542 .0664538
	_cons	5.044056	.1999372	25.23	0.000 5.435926
dy	age	.0879359	.0278018	3.16	0.002 .1424264
	female	.6626649	.061461	10.78	0.000 .7831263
	educ	.0619485	.0113921	5.44	0.000 .0842766
	blhisp	-.3639377	.0612186	-5.94	0.000 -.2439515
	totchr	.7969518	.0736115	10.83	0.000 .9412276
	ins	.1701367	.061304	2.78	0.006 .2902904
	income	.0027078	.0013265	2.04	0.041 .0053077
/athrho	_cons	-.6760546	.1873167	-3.61	0.000 -.3089207
	/lnsigma	-.1313456	.1029162	-1.28	0.202 .0703664
	/lnsigma	.2398173	.0152203	15.76	0.000 .2696485
	rho	-.1305955	.1011609		-.3212655 .0702505
	sigma	1.271017	.0193452		1.233661 1.309504
	lambda	-.1659891	.128968		-.4187617 .0867836

Wald test of indep. eqns. (rho = 0): chi2(1) = 1.63 Prob > chi2 = 0.2019

The results are only slightly different from those reported above, although `income` appears to have significant additional explanatory power. Furthermore, the use of this exclusion restriction is debatable because there are reasons to expect that `income` should also appear in the outcome equation.

Another command that delivers output almost identical to that produced by the `heckman` ML command is Stata's `eregress` command, which has

syntax almost identical to that of the former, as can be seen below (but output is suppressed):

```
* Heckman MLE with exclusion restriction
. eregress lny $xlist, select(dy = $xlist income) nolog vce(robust)
(output omitted)
```

19.7 Nonnormal models of selection

The classical fully parametric approach to estimation of selection model has limitations. First, controlling for selection relies heavily on the distributional assumption of joint normality. Second, it typically requires that we have available a nontrivial excluded exogenous variable that serves as an identifying restriction. Further, its test of selection bias relies on a linear measure of dependence (correlation) between the unobserved factors that simultaneously impact both selection and the outcome.

In principle, one can generalize the selection model to any bivariate distribution, but in practice there are relatively few suitable bivariate distributions. For example, the bivariate normal is unusual in its tractability and in its properties of having conditional distributions and marginal distributions that are normally distributed.

19.7.1 Copula models

These limitations can be addressed using a copula-based estimator. This permits choice of marginal distributions for the two parts of the selection model that are well-suited to the type of data being analyzed and then uses a copula function to introduce correlation or, more generally, dependence, between these two parts.

This fully parametric approach is founded on Sklar's Theorem, which proves that under certain conditions, marginal distributions of variables, say, $f(y_1|\beta_1)$ and $g(y_2|\beta_2)$, can be combined using a copula function to obtain a joint distribution $h(y_1, y_2|\beta_1, \beta_2, \theta)$, where θ is a scalar measure of dependence between y_1 and y_2 . The properties of the dependence parameter vary with the functional form of the copula used, and there are many potential copulas with varying degrees of restrictiveness. Given the choice of functional forms of marginals and the copula, ML methods can be used to estimate all the parameters.

The first step of this approach is to specify the marginal distributions (which will be conditioned on exogenous variables \mathbf{z}) for the latent variable that determines the selection indicator and for the latent outcome variable. A logistic (logit) or normal (probit) is the obvious choice of distribution for the selection latent variable, while the distribution for the latent outcome will vary with the type of data.

The second step is to use the copula function that binds these two densities. Let $f(y_1^*|\mathbf{z}, \beta_1)$ and $g(y_2^*|\mathbf{z}, \beta_2)$ denote the densities for, respectively, the selection equation latent variable and the outcome latent variable. These are combined using a specified parametric copula, denoted $C\{f(y_1^*|\mathbf{z}, \beta_1), g(y_2^*|\mathbf{z}, \beta_2), \theta\}$, where θ is a scalar-valued dependence parameter. This generates a joint distribution in latent variables that leads to a tractable joint density for the observed data (d, y_2) that can be estimated by ML. This method separates the specification and estimation of the marginal from the problem of inference about the dependence parameter.

The flexibility of the approach comes from the feasibility of varying both the marginal distributions and the functional form of the copula, $C(\cdot)$. Further, provided the margins are sufficiently flexible, this approach potentially can model a variety of symmetric and asymmetric dependence structures, not just linear dependence as under a normal distribution. A broader type of sample-selection effect can be captured.

Table 19.2 presents the functional forms of four widely used bivariate copulas for uniform random variables (u_1, u_2) . Different copulas have different domains and place different restrictions on the dependence parameter. Some restrict dependence to be nonnegative. And the special case of independence does not always correspond to $\theta = 0$. For example, the Gumbel copula implies independence if $\theta = 1$.

Table 19.2. Some standard copula functions

Copula type	Function $C(u_1, u_2)$	θ -domain
Gaussian	$\Phi_G[\Phi_G^{-1}(u_1)\Phi_G^{-1}(u_2); \theta]$	$-1 < \theta < +1$
Student's t	$\int_{-\infty}^{t_{\nu}^{-1}(u_1)} \int_{-\infty}^{t_{\nu}^{-1}(u_2)} \frac{1}{2\pi(1-\theta^2)^{1/2}} \times \left\{ 1 + \frac{(s^2 - 2\theta st + t^2)}{\nu(1-\theta^2)} \right\}^{-(\nu+2)/2} ds dt$	$-1 < \theta < +1$
Clayton	$(u_1^{-\theta} + xu_2^{-\theta} - 1)^{-1/\theta}$	$(0, \infty)$
Gumbel	$\exp(-(\tilde{u}_1^\theta + \tilde{u}_2^\theta)^{1/\theta})$, $\tilde{u}_j = -\ln u_j$	$[1, \infty)$

Having to choose the functional form of the copula is a potential disadvantage. In practice, one may consider several combinations of copulas and marginal distributions and select the “best” model according to an information criterion.

19.7.2 Copula application

The community-contributed `heckmancopula` command ([Hasebe 2013](#)) is structured to fit the classic Heckman selection model with a wide range of distributions. The latent variable for the selection equation may be `normal` or `logistic`. The latent outcome variable may be `normal`, `logistic`, or Student’s t . The copula may be `product`, `gaussian`, `fgm`, `plackett`, `amh`, `clayton`, `frank`, `gumbel`, or `joe`.

We first present results for the Gaussian copula with normal marginals. This duplicates ML estimation of the standard selection model using the `heckman` command.

```

. * Copula-based selection models with Gaussian copula (same as heckman)
. qui use mus219mepsambexp, clear
. global xlist age female educ blhisp totchr ins // Regressor list $xlist
. heckmancopula lny $xlist, select(dy = $xlist income) copula(gaussian)
>     margsel(probit) margin1(normal) vce(robust)

Iteration 0:  log pseudolikelihood = -5836.7639
Iteration 1:  log pseudolikelihood = -5836.2404
Iteration 2:  log pseudolikelihood = -5836.2193
Iteration 3:  log pseudolikelihood = -5836.2192

```

Sample Selection Model: Copula gaussian, Margins probit-normal

Number of obs = 3,328
 Wald chi2(7) = 325.56
 Prob > chi2 = 0.0000

Log pseudolikelihood = -5836.2192

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
select					
age	.0879359	.0278018	3.16	0.002	.0334453 .1424264
female	.6626649	.061461	10.78	0.000	.5422036 .7831263
educ	.0619485	.0113921	5.44	0.000	.0396203 .0842766
blhisp	-.3639377	.0612186	-5.94	0.000	-.4839239 -.2439515
totchr	.7969518	.0736115	10.83	0.000	.652676 .9412277
ins	.1701368	.061304	2.78	0.006	.0499831 .2902904
income	.0027078	.0013265	2.04	0.041	.0001078 .0053077
_cons	-.6760546	.1873167	-3.61	0.000	-1.043189 -.3089207
lny					
age	.2119749	.0225429	9.40	0.000	.1677915 .2561582
female	.3481439	.0541676	6.43	0.000	.2419774 .4543104
educ	.018716	.0099997	1.87	0.061	-.0008831 .038315
blhisp	-.2185714	.0578622	-3.78	0.000	-.3319791 -.1051636
totchr	.5399199	.030624	17.63	0.000	.4798979 .5999418
ins	-.0299872	.0492054	-0.61	0.542	-.1264281 .0664537
_cons	5.044057	.1999372	25.23	0.000	4.652187 5.435927
lnsigma					
_cons	.2398173	.0152203	15.76	0.000	.2099862 .2696485
atheta					
_cons	-.1313461	.1029162	-1.28	0.202	-.3330582 .0703661
theta					
tau	-.1305959	.101161			-.321266 .0702502
					-.0447595 .2082167

Wald test of independence : Test statistic 1.667 with p-value 0.1967

. estimates store Gaussian_N

The output is identical to that earlier for the `heckman` example with exclusion restriction. The parameter `theta` is the Pearson correlation coefficient and is not significantly different from zero by either the Wald test or the Lagrange multiplier test. The parameter `tau` reports Kendall's tau, an alternative measure of correlation to the Pearson correlation coefficient. There is little evidence of significant selection.

Next, we fit three additional nonnormal selection models. All use probit for the selection equation. All but the last use the normal for the selection equation. The Frank, Farlie–Gumbel–Morgenstern, and Plackett copulas are used.

```

. * Copula-based selection models with several different copulas
. qui heckmancopula lny $xlist, select(dy = $xlist income) copula(frank)
>      margsel(probit) margin1(normal) vce(robust)
. estimates store Frank_N
. qui heckmancopula lny $xlist, select(dy = $xlist income)
>      copula(fgm) margsel(probit) margin1(normal) vce(robust)
. estimates store FGM_N
. qui heckmancopula lny $xlist, select(dy = $xlist income)
>      copula(plackett) margsel(probit) margin1(t) df(10) vce(robust)
. estimates store Plackett_t10
. estimates table Gaussian_N Frank_N FGM_N Plackett_t10,
>      eq(1) se b(%13.4f) stats(N ll)

```

Variable	Gaussian_N	Frank_N	FGM_N	Plackett_t10
#1				
age	0.0879 0.0278	0.0868 0.0278	0.0869 0.0281	0.0885 0.0279
female	0.6627 0.0615	0.6635 0.0615	0.6635 0.0615	0.6641 0.0614
educ	0.0619 0.0114	0.0619 0.0114	0.0619 0.0114	0.0618 0.0114
blhisp	-0.3639 0.0612	-0.3658 0.0612	-0.3657 0.0613	-0.3637 0.0612
totchr	0.7970 0.0736	0.7957 0.0738	0.7959 0.0741	0.7982 0.0736
ins	0.1701 0.0613	0.1691 0.0614	0.1691 0.0614	0.1700 0.0613
income	0.0027 0.0013	0.0027 0.0013	0.0027 0.0013	0.0027 0.0013
_cons	-0.6761 0.1873	-0.6684 0.1872	-0.6690 0.1888	-0.6774 0.1878
lny				
age	0.2120 0.0225	0.2173 0.0221	0.2171 0.0229	0.2164 0.0223
female	0.3481 0.0542	0.3798 0.0489	0.3786 0.0553	0.3621 0.0498
educ	0.0187 0.0100	0.0223 0.0097	0.0222 0.0098	0.0244 0.0099
blhisp	-0.2186 0.0579	-0.2388 0.0560	-0.2380 0.0584	-0.2259 0.0561
totchr	0.5399 0.0306	0.5621 0.0282	0.5613 0.0310	0.5509 0.0278
ins	-0.0300 0.0492	-0.0207 0.0487	-0.0210 0.0489	-0.0363 0.0483
_cons	5.0441 0.1999	4.9061 0.1716	4.9109 0.1993	4.9589 0.1800
lnsigma				
_cons	0.2398 0.0152	0.2374 0.0151	0.2374 0.0151	0.1381 0.0145
atheta				
_cons	-0.1313 0.1029	0.0098 0.0004	-0.0089 0.3229	-0.2924 0.1563
Statistics				
N	3328	3328	3328	3328
ll	-5836.2192	-5836.6736	-5836.6729	-5827.7743

Legend: b/se

All models use the same number of parameters, so we can directly compare log likelihoods. The first three models give similar log likelihoods, which is not surprising, because there appears to be little evidence of sample selection for these data. The final model fits considerably better because we use a t distribution rather than the normal distribution for the outcome; the t distribution has heavier tails than the normal.

19.8 Prediction from models with outcome in logs

For the models considered in this chapter, conditional prediction is an important application of the estimated parameters of the model. Such an exercise may be of the within-sample type, or it may involve comparison of fitted values under alternative scenarios, as illustrated in section 4.3.

Whether a model predicts well within the sample is obviously an important consideration in model comparison and selection.

Calculation and comparison of predicted values is relatively simpler in the levels form of the model because there is no retransformation involved. In the current analysis, the dependent variable is log transformed, but one wants predictions in levels, and hence the retransformation problem, first mentioned in section 4.2.3, must be confronted.

Table 19.3 provides expressions for the conditional and unconditional means for the three models with outcome in logs rather than levels, presented in sections 19.4–19.6. The predictors are functions that depend upon the linear-index function, $x'\beta$, and variance and covariance parameters, σ^2 , σ_2^2 , and σ_{12} . These formulas are derived under the twin assumptions of normality and homoskedasticity. The dependence of the predictor on variances estimated under the assumption of homoskedastic errors is potentially problematic for all three models because if that assumption is incorrect, then the usual estimators of variance and covariance parameters will be biased.

Table 19.3. Conditional and unconditional means for models in logs

Moment	Model	Prediction function
$E(y \mathbf{x}, y > 0)$	Tobit	$\exp(\mathbf{x}'\boldsymbol{\beta} + \sigma^2/2)[1 - \Phi\{(\gamma - \mathbf{x}'\boldsymbol{\beta})/\sigma\}]^{-1}$ $[1 - \Phi\{(\gamma - \mathbf{x}'\boldsymbol{\beta} - \sigma^2)/\sigma\}]$
$E(y \mathbf{x})$	Tobit	$\exp(\mathbf{x}'\boldsymbol{\beta} + \sigma^2/2)[1 - \Phi\{(\gamma - \mathbf{x}'\boldsymbol{\beta} - \sigma^2)/\sigma\}]$
$E(y_2 \mathbf{x}, y_2 > 0)$	Two-part	$\exp(\mathbf{x}'_2\boldsymbol{\beta}_2 + \sigma_2^2/2)$
$E(y_2 \mathbf{x})$	Two-part	$\exp(\mathbf{x}'_2\boldsymbol{\beta}_2 + \sigma_2^2/2)\Phi(\mathbf{x}'_1\boldsymbol{\beta}_1)$
$E(y_2 \mathbf{x}, y_2 > 0)$	Selection	$\exp(\mathbf{x}'_2\boldsymbol{\beta}_2 + \sigma_2^2/2)\{1 - \Phi(-\mathbf{x}'_1\boldsymbol{\beta}_1)\}^{-1}$ $\{1 - \Phi(-\mathbf{x}'_1\boldsymbol{\beta}_1 - \sigma_{12}^2)\}$
$E(y_2 \mathbf{x})$	Selection	$\exp(\mathbf{x}'_2\boldsymbol{\beta}_2 + \sigma_2^2/2)\{1 - \Phi(-\mathbf{x}'_1\boldsymbol{\beta}_1 - \sigma_{12}^2)\}$

19.8.1 Predictions from tobit in logs

We begin by fitting $E(y|\mathbf{x})$ and $E(y|\mathbf{x}, y > 0)$ for the tobit model in logs.

```

. * Prediction from tobit regression with dependent variable lny
. qui use mus219mepsambexp, clear
. scalar gamma = -0.0000001
. qui tobit lny $xlist, ll(gamma) vce(robust)
. predict xb, xb                                // xb is estimate of x'b
. matrix btobit = e(b)
. scalar sigma = sqrt(btobit[1,e(df_m)+2]) // sigma is estimate of sigma
. generate threshold = (gamma-xb)/sigma      // gamma: lower censoring point
. generate yhat = exp(xb+0.5*sigma^2)*(1-normal((gamma-xb-sigma^2)/sigma))
. generate ytrunchat = yhat / (1 - normal(threshold)) if dy==1
(526 missing values generated)
. summarize y yhat

```

Variable	Obs	Mean	Std. dev.	Min	Max
y	3,328	1386.519	2530.406	0	49960
yhat	3,328	45805.92	273444.7	133.9768	1.09e+07

```
. summarize y yhat ytrunchat if dy==1
```

Variable	Obs	Mean	Std. dev.	Min	Max
y	2,802	1646.8	2678.914	1	49960
yhat	2,802	53271.51	297386.4	283.4537	1.09e+07
ytrunchat	2,802	53536.85	297376.6	383.6245	1.09e+07

The estimates, denoted by `yhat` and `ytrunchat`, confirm that these predictors are very poor. Mean expenditure is overpredicted in both cases and more so in the censored case. The reported results reflect the high sensitivity to estimates of σ^2 , as $\exp(\sigma^2/2)$ appears multiplicatively in table 19.3. The tobit model has $\hat{\sigma}^2 = 7.735$, whereas from section 19.5.3 the two-part model has a much, much lower $\hat{\sigma}^2 = 1.2696^2 = 1.612$.

19.8.2 Predictions from two-part model in logs

Predictions of $E(y_2|\mathbf{x})$ and $E(y_2|\mathbf{x}, y_2 > 0)$ from the two-part model are considerably better but still biased. We first transform the fitted log values from the conditional part of the two-part model, assuming normality.

```

. * Two-part model predictions
. qui probit dy $xlist
. predict dyhat, pr
. qui regress lny $xlist if dy==1
. predict x xpos, xb
. generate yhatpos = exp(xpos+0.5*e(rmse)^2)

```

Next, we generate an estimate of the unconditional values, denoted by `yhat2step`, by multiplying by the fitted probability of the positive expenditure `dyhat` from the probit regression.

```

. * Unconditional prediction from two-part model
. generate yhat2step = dyhat*yhatpos
. summarize yhat2step y

```

Variable	Obs	Mean	Std. dev.	Min	Max
yhat2step	3,328	1680.978	2012.084	87.29432	40289.03
y	3,328	1386.519	2530.406	0	49960

Variable	Obs	Mean	Std. dev.	Min	Max
yhatpos	2,802	1995.981	2087.072	430.8354	40289.03
y	2,802	1646.8	2678.914	1	49960

The mean of the predicted values is considerably closer to the sample average than to those for the tobit estimator, confirming the greater robustness of the two-part model.

19.8.3 Predictions from selection model

Finally, we predict $E(y_2|\mathbf{x})$ and $E(y_2|\mathbf{x}, y_2 > 0)$ for the selection model.

```

. * Heckman model predictions
. qui heckman lny $xlist, select(dy = $xlist) vce(robust)
. predict probpos, psel
. predict x1b1, xbsel
. predict x2b2, xb
. scalar sig2sq = e(sigma)^2
. scalar sig12sq = e(rho)*e(sigma)^2
. display "sigma1sq = 1" " sigma12sq = " sig12sq " sigma2sq = " sig2sq
sigma1sq = 1 sigma12sq = -.20055906 sigma2sq = 1.6147766
. generate yhatheck = exp(x2b2 + 0.5*(sig2sq))*(1 - normal(-x1b1-sig12sq))
. generate yhatposheck = yhatheck/probpos
. summarize yhatheck y probpos dy

```

Variable	Obs	Mean	Std. dev.	Min	Max
yhatheck	3,328	1659.802	1937.095	74.32413	37130.18
y	3,328	1386.519	2530.406	0	49960
probpos	3,328	.8415738	.1411497	.2029135	1
dy	3,328	.8419471	.3648454	0	1

```
. summarize yhatposheck probpos dy y if dy==1
```

Variable	Obs	Mean	Std. dev.	Min	Max
yhatposheck	2,802	1970.923	2003.406	389.4755	37130.18
probpos	2,802	.8661997	.1237323	.2867923	1
dy	2,802	1	0	1	1
y	2,802	1646.8	2678.914	1	49960

The predictions from the selection model, denoted by `yhatheck`, are close to those from the two-part model. The main difference from the two-part model comes from introducing correlation in the errors, and here the correlation of – 0.124 is low.

The poor prediction performance of the original tobit model confirms the earlier conclusions about its unsuitability for modeling the current dataset.

19.9 Endogenous regressors

The preceding analysis applies when regressors are exogenous. When instead some regressors are endogenous, standard estimators such as the tobit MLE become inconsistent, and alternative estimators are needed. The following methods that control for endogeneity rely on very strong distributional assumptions.

19.9.1 Tobit model with endogenous regressors

Endogenous regressors in the tobit model can be accommodated by using a structural model approach that is similar to that for the probit model presented in section [17.9](#). [Cameron and Trivedi \(2005\)](#), chap. 16.8) and especially [Wooldridge \(2010\)](#), chap. 17.5) provide details.

If there is only one endogenous regressor, then the setup involves two equations—the structural equation of interest and the reduced form for the endogenous regressor. The structural model is exactly the same as [\(17.9\)](#) and [\(17.10\)](#), except now we observe $y_{1i} = y_{1i}^*$ if, for example, $y_{1i}^* > 0$. The framework assumes that the endogenous regressor is continuous, so the method should not be used for a discrete endogenous variable. The reduced-form equation for this variable must include at least one exogenous instrumental variable that affects the outcome variable only through the endogenous regressor and so is excluded from the structural equation. Error terms are assumed to be joint normally distributed.

The `ivtobit` command is similar to the `ivprobit` command and computes both the MLE and the computationally simpler but less efficient two-step estimator of [Newey \(1987\)](#). The `ivtobit` command has options for ME, prediction, and variance estimation similar to those for the `tobit` command.

If instruments are weakly correlated with the endogenous regressor, then the usual asymptotic theory may perform poorly. Then alternative methods may be used. Inference for weak instruments for the linear model is presented in section 7.7. The discussion there includes methods based on

minimum distance estimation due to [Magnusson \(2010\)](#) that can be applied to a wide range of linear and nonlinear structural models. The community-contributed `rivtest` ([Finlay and Magnusson 2009](#)) and `weakiv` programs ([Finlay, Magnusson, and Schaffer 2014](#)) apply these methods following estimation using `ivtobit`.

19.9.2 Richer models with endogenous regressors

The `eintreg` and `eregress` commands provide ML estimates of a wide range of models with endogenous regressors or sample selection, or both. Furthermore, endogenous variables can be continuous, binary, and ordinal.

These commands are members of the class of ERM commands for recursive models such as [\(17.9\)](#) and [\(17.10\)](#) with error terms that are joint normal distributed; see section [23.7](#) for details.

19.9.3 Endogenous-switching regression model

The sample-selection model leads to an observed outcome if the selection indicator $y_1 > 0$ and not observed otherwise. The endogenous-switching regression model instead specifies that one value of the outcome is observed if $y_1 > 0$ and a different value is observed if $y_1 \leq 0$. For example, the wage may vary according to whether a worker is a union worker.

Then we observe

$$\begin{aligned} y_2 &= \mathbf{x}'_2 \boldsymbol{\beta}_2 + \varepsilon_2, & \text{if } y_1 > 0 \\ y_3 &= \mathbf{x}'_3 \boldsymbol{\beta}_3 + \varepsilon_3, & \text{if } y_1 \leq 0 \end{aligned}$$

where ε_2 and ε_3 are possibly correlated with the selection equation error ε_1 .

Several community-contributed commands fit this model, including the command `heckmancopula`.

19.10 Missing data

Truncated regression models and sample-selection models are examples of models developed to handle missing data. (Censored regression such as for top-coded data, by contrast, has data that are incompletely observed rather than missing.)

In this section, we provide a general discussion of various ways that data may be missing, the consequences (if any) of this missingness, and how to control for missing data. [Wooldridge \(2010, chap. 19\)](#) covers various selection mechanisms and methods to control for selection, most notably, inverse-probability weighting (IPW) and Heckman-type selection models. [Cameron and Trivedi \(2005, chap. 27\)](#) include coverage of multiple imputation and the statistics literature on missing-data mechanisms. [Seaman and White \(2013\)](#) provide a useful survey of inverse-probability weighting and contrast it with multiple imputation.

This section focuses on cross-sectional data, while the following section considers application to panel-data attrition. Missing-data methods are also used in the treatment evaluation chapters [24–25](#) and in sections [30.5–30.6](#).

19.10.1 Missing-data mechanisms

The simplest case of missing data is where the cause of missing data is unrelated to either observed data or missing data. This case is referred to as data being missing completely at random (MCAR). In this case, there is no problem, and we can do the usual statistical analysis on just those observations for which complete data are available. For example, we drop from the analysis any observations for which data are incomplete. The only downside is a loss of estimator precision due to less data.

If data are instead not MCAR, there are many different missing-data mechanisms and remedies and related terminology that differs across various branches of statistics.

One set of terminology breaks data that are not MCAR into either missing at random (MAR) or missing not at random (MNAR). Missingness is MAR if the cause of the missing data is unrelated to missing data though is related to observed data. Missingness is MNAR if the cause of the missing data is related to missing data.

For regression analysis, where distinction is made between endogenous variables and exogenous variables, a natural distinction is that between selection on observables only and selection on unobservables.

Under selection on observables only, the missingness mechanism is purely random, after appropriate adjustment for exogenous variables. This is a conditional form of MAR that is also described as exogenous sample selection, ignorable selection (conditional on exogenous variables), or, in the treatment-effects literature, unconfoundedness.

Under selection on unobservables, even after one controls for exogenous variables, the missingness mechanism depends on unobservables that are not independent of the outcome variable y . Then data are MNAR, a situation also described as endogenous sample selection or nonignorable selection (even after conditioning on exogenous variables). Much stronger assumptions are then needed to obtain consistent estimates given the selected sample.

19.10.2 Complete-case analysis

Consider the linear regression model $y_i = \mathbf{x}'_i \boldsymbol{\beta} + u_i$ and the OLS estimator $\hat{\boldsymbol{\beta}}$ with first-order conditions $\sum_{i=1}^N \mathbf{x}_i(y_i - \mathbf{x}'_i \boldsymbol{\beta}) = \mathbf{0}$. The essential condition for consistency is that $E(\mathbf{x}_i u_i) = \mathbf{0}$. A sufficient, though not necessary, condition for $E(\mathbf{x}_i u_i) = \mathbf{0}$ is that $E(u_i | \mathbf{x}_i) = 0$.

Complete-case analysis, also called case deletion or listwise deletion, performs OLS estimation using only those observations for which complete data are available.

Introduce the selection indicator s_i , which equals 1 if complete data on (y_i, \mathbf{x}_i) are available and equals 0 otherwise. The OLS estimator $\hat{\boldsymbol{\beta}}_{CC}$ using

only available data (complete-case analysis) solves the first-order conditions

$$\sum_{i=1}^N s_i \mathbf{x}_i (y_i - \mathbf{x}'_i \boldsymbol{\beta}) = \mathbf{0}$$

because only nonmissing data (with $s_i = 1$) are included in this regression. The key assumption for consistency of $\hat{\boldsymbol{\beta}}_{CC}$ is then

$$E(s_i \mathbf{x}_i u_i) = \mathbf{0} \quad (19.8)$$

Condition (19.8) holds under MCAR because if missingness is purely random, then $E(s_i \mathbf{x}_i u_i) = E(s_i) \times E(\mathbf{x}_i u_i) = E(s_i) \times \mathbf{0} = \mathbf{0}$, under the standard OLS assumption that $E(\mathbf{x}_i u_i) = \mathbf{0}$.

Condition (19.8) also holds under a conditional mean version of MAR that missingness is related only to the regressors \mathbf{x}_i and not to the error term u_i . Specifically, we suppose that $E(u_i | \mathbf{x}_i, s_i) = 0$. Then $E(s_i \mathbf{x}_i u_i) = E_{\mathbf{x}, s} \{E(u_i | \mathbf{x}_i, s_i)\} = E_{\mathbf{x}, s}(0) = 0$. A sufficient condition for $E(u_i | \mathbf{x}_i, s_i) = 0$ is that 1) $E(u_i | \mathbf{x}_i) = 0$, a stronger condition than $\text{Cov}(\mathbf{x}_i, u_i) = \mathbf{0}$; and 2) $s_i = h(\mathbf{x}_i)$ for some function $h(\cdot)$. Then $E(u_i | \mathbf{x}_i, s_i) = E\{u_i | \mathbf{x}_i, h(\mathbf{x}_i)\} = E(u_i | \mathbf{x}) = 0$.

This analysis carries over more generally to the usual m -estimators such as the MLE and to IV estimators; in the latter case, sufficient conditions are that $E(u_i | \mathbf{z}_i) = 0$ and $s_i = h(\mathbf{z}_i)$, where \mathbf{z}_i are the instruments.

In summary, the commonly used practice of restricting analysis to only those observations with complete data is fine if 1) missingness is completely at random; or 2) if missingness is related only to included exogenous regressors and the model is correctly specified, for example, $E(u_i | \mathbf{x}_i) = 0$ in the case of OLS.

19.10.3 Inverse probability weighting

Complete-case analysis leads to inconsistent OLS estimation if $E(u_i|\mathbf{x}_i, s_i) \neq 0$ or, equivalently, if $E(y_i|\mathbf{x}_i, s_i) \neq \mathbf{x}'_i\boldsymbol{\beta}$. Under some assumptions, a weighted least-squares (LS) estimator is consistent.

We present results for the more general case of an m -estimator $\hat{\boldsymbol{\theta}}$ that solves

$$\sum_{i=1}^N \mathbf{q}(\mathbf{w}_i, \boldsymbol{\theta}) = \mathbf{0}$$

where $\mathbf{w}_i = (y_i, \mathbf{x}_i)$ and, for example, $\mathbf{q}(\mathbf{w}_i, \boldsymbol{\theta}) = \mathbf{x}_i(y_i - \mathbf{x}'_i\boldsymbol{\theta})$ for OLS. The essential condition for consistency is that $E\{\mathbf{q}(\mathbf{w}_i, \boldsymbol{\theta})\} = \mathbf{0}$.

The complete-case estimator $\hat{\boldsymbol{\theta}}_{cc}$ solves $\sum_{i=1}^N s_i \mathbf{q}(\mathbf{w}_i, \boldsymbol{\theta}) = \mathbf{0}$. The essential condition for consistency is that $E\{s_i \mathbf{q}(\mathbf{w}_i, \boldsymbol{\theta})\} = \mathbf{0}$. We are concerned with situations where this condition does not hold, because selection s_i is not completely random or is not only determined by the exogenous regressors \mathbf{x}_i .

The key is to control for selection by introducing additional exogenous variables beyond the included regressors \mathbf{x}_i . Let \mathbf{z}_i be a set of variables that does not include y_i but can include variables in \mathbf{x}_i , and let the probability that data are nonmissing be

$$p(\mathbf{z}_i) = \Pr(s_i = 1 | \mathbf{z}_i)$$

where $0 < p(\mathbf{z}_i) < 1$. Then the IPW estimator $\hat{\boldsymbol{\theta}}_{IPW}$ solves

$$\sum_{i=1}^N \frac{s_i}{p(\mathbf{z}_i)} \mathbf{q}(\mathbf{w}_i, \boldsymbol{\theta}) = \mathbf{0}$$

For $\hat{\boldsymbol{\theta}}_{\text{IPW}}$ to be consistent, we need $E\{s_i \mathbf{q}(\mathbf{w}_i, \boldsymbol{\theta})/p(\mathbf{z}_i)\} = \mathbf{0}$. In general,

$$\begin{aligned} E_{s, \mathbf{w}, \mathbf{z}}\{s \mathbf{q}(\mathbf{w})/p(\mathbf{z})\} &= E_{\mathbf{w}, \mathbf{z}}[E_{s|\mathbf{w}, \mathbf{z}}\{s \mathbf{q}(\mathbf{w})/p(\mathbf{z})\} | \mathbf{w}, \mathbf{z}] \\ &= E_{\mathbf{w}, \mathbf{z}}\{E_{s|\mathbf{w}, \mathbf{z}}(s|\mathbf{w}, \mathbf{z}) \times \mathbf{q}(\mathbf{w})/p(\mathbf{z})\} \\ &= E_{\mathbf{w}, \mathbf{z}}\{\Pr(s = 1|\mathbf{w}, \mathbf{z}) \times \mathbf{q}(\mathbf{w})/p(\mathbf{z})\} \end{aligned}$$

where the last line uses the fact that for a $(0, 1)$ random variable, $E(s) = \Pr(s = 1)$. It follows that a sufficient condition is that

$$\Pr(s_i = 1|\mathbf{w}_i, \mathbf{z}_i) = \Pr(s_i = 1|\mathbf{z}_i)$$

because then

$$\begin{aligned} E_{\mathbf{w}, \mathbf{z}}\{\Pr(s = 1|\mathbf{w}, \mathbf{z}) \mathbf{q}(\mathbf{w})/p(\mathbf{z})\} &= E_{\mathbf{w}, \mathbf{z}}\{p(\mathbf{z}) \mathbf{q}(\mathbf{w})/p(\mathbf{z})\} \\ &= E_{\mathbf{w}, \mathbf{z}}\{\mathbf{q}(\mathbf{w})\} \\ &= \mathbf{0} \end{aligned}$$

where the last line uses the original assumption that $E\{\mathbf{q}(\mathbf{w}_i, \boldsymbol{\theta})\} = \mathbf{0}$.

Implementation of the IPW estimator requires the weights $p(\mathbf{z}_i)$. Sample survey data may directly provide the weights $p(\mathbf{z}_i)$; see section 6.9.

More often, the weights need to be estimated. A common approach is to specify and fit a logit (or probit) model for whether $s_i = 1$. The weights then are $p(\mathbf{z}_i, \hat{\boldsymbol{\alpha}})$, where $\hat{\boldsymbol{\alpha}}$ are logit parameter estimates. Strictly speaking, inference on $\hat{\boldsymbol{\theta}}_{\text{IPW}}$ should then control for the first-step estimation of $\boldsymbol{\alpha}$; for example, one could bootstrap the two-step estimator. It has been shown that it is okay to ignore this complication and use the usual robust standard errors (heteroskedastic robust for independent data or cluster-robust for clustered data) because this leads, surprisingly, to overestimation of the standard error of $\hat{\boldsymbol{\theta}}_{\text{IPW}}$ and consequent conservative inference.

The big practical issue is determining the variables to include in \mathbf{z}_i . In some cases, the only missing data are those on y_i for some observations, with \mathbf{x}_i only observed. Then we let $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{v}_i)$, where we desire additional variables \mathbf{v}_i that do not belong in the original model of interest but do make more credible the assumption that

$\Pr(s_i = 1|y_i, \mathbf{x}_i, \mathbf{v}_i) = \Pr(s_i = 1|\mathbf{x}_i, \mathbf{v}_i)$. For panel attrition, the pair $(y_{it}, \mathbf{x}_{it})$ may be missing for some observations. Then for pooled estimators such as pooled OLS we need \mathbf{z}_{it} such that

$\Pr(s_{it} = 1|y_{it}, \mathbf{x}_{it}, \mathbf{z}_{it}) = \Pr(s_{it} = 1|\mathbf{z}_{it})$. Such assumptions are called selection on observables only; once we condition on a large enough set of exogenous variables, selection is no longer related to the endogenous variable y .

19.10.4 Endogenous sample selection

The IPW estimator expands the set of exogenous variables, so that conditional on this expanded set, the data are satisfy the MAR assumption.

Endogenous sample selection arises when data on some observations are missing in part because of values of endogenous variables even after controlling for exogenous variables. A simple example is truncation at zero where we observe only (y_i, \mathbf{x}_i) if $y_i > 0$; see section [19.3.5](#).

A more complicated example is the selection model of section [19.6](#), where we may observe y_i if a related variable crosses a threshold. We apply this model to panel data with attrition in section [19.11](#). Other endogenous sample-selection models include the endogenous-switching regression model of section [19.9.3](#).

Endogenous selection models require much stronger parametric assumptions about error terms than those necessary to hold for exogenous selection. Thus, wherever justifiable, the assumption of selection on observables only is made. This allows use of methods that, conditional on this strong assumption, rely on fewer parametric assumptions.

19.10.5 Imputation

Complete-case analysis and IPW lead to complete loss of an observation whenever any part of (y_i, \mathbf{x}_i) is missing. When only data on some variables for some observations are missing, it is tempting to impute the missing data to create a complete observation.

Care is needed, however. If the case-deletion estimate is inconsistent, then so will be estimates obtained using the simplest imputation methods. For example, suppose that we simply want to estimate $E(y)$, data on some observations are missing, and the case-deletion estimate \bar{y}_{CC} is inconsistent for $E(y)$ because of nonrandom selection. Then replacing each missing observation on y_i with the mean \bar{y}_{CC} yields a sample average that again equals the inconsistent estimate \bar{y}_{CC} . Furthermore, even if there was no selection problem, the apparent increase in sample size will lead to an artificially lowered standard error of the estimate.

[Abrevaya and Donald \(2017\)](#) propose a GMM method to obtain more efficient estimates for linear regression with independent heteroskedastic errors when there are missing observations on exogenous or endogenous regressors, under a variation of MAR suitable to their application. While this is a quite specialized setting, it arises often in applied microeconomics studies.

Multiple-imputation methods specify a stochastic model for the missing data as a function of the observed data. Random draws of the missing data are generated, rather than a deterministic value such as a predicted mean. Estimation is then based on several completed datasets composed of nonmissing data where available and imputed data otherwise. For details, see section [30.5](#).

19.11 Panel attrition

We present panel attrition here rather than in the panel chapters because it uses methods presented in sections [19.6](#) and [19.10](#). Readers already familiar with this material can tackle the current section as an immediate extension of chapter 8. Going the other way, readers of the current chapter may need to revisit chapter 8 to refresh familiarity with panel commands.

For a variety of reasons, the participants in a survey may choose to not participate at all (survey nonresponse) or may choose to not answer particular questions in a survey (item nonresponse), sometimes because the questions address issues that respondents may consider sensitive. The resulting sample is said to be unbalanced because it leads to a different number of observations across sampled respondents; see [Baltagi and Song \(2006\)](#). Unbalanced panel samples are quite common.

Panel attrition refers to loss of data in successive panel surveys that occurs because some of the original survey participants drop out of the survey at various stages. In some cases, the nonrespondent may permanently exit from the survey; in other cases, the nonrespondent may return to the sample in a later period.

This section considers the consequences of ignoring sample attrition and the statistical remedies available to deal with the problem. The simplest approach is to assume that attrition is a completely random and ignorable process unrelated to either observed or unobserved characteristics of the respondents. More complicated approaches add a regression model that characterizes the attrition (or selection) process as a function of observable characteristics of respondents and, if observables alone cannot control for attrition, unobservable characteristics of respondents. This leads to a two-equation model like the one considered in section [19.6](#).

Panel data add an additional dimension that is not present in the standard missing-data model. In multiperiod panels, the extent of missing data may grow as additional waves are added. For example, for the Michigan Panel Study of Income Dynamics, nearly 50% of the initial sample of 1969 was lost through attrition by 1989. One approach to deal with this issue is to use

sample refreshment that adds additional survey respondents in later waves of the sample survey to adjust for the missing data from earlier waves.

We focus on implementation of methods for panel attrition, along with a brief exposition of the underlying theory. A careful statement of assumptions used for a range of corrections for panel attrition is given in [Wooldridge \(2010, chap. 19.9\)](#).

19.11.1 Attrition empirical example

We use data from the first three waves of the Australian Medicine in Australia Balancing Employment and Life survey. This survey, introduced in section 4.6.2, collects data on doctors' earnings.

Specifically, we analyze log annual earnings (`lyearn`) of female general practitioners as a function of log annual hours worked, years of experience, squared years of experience, and an indicator of hospital work, which are variables labeled `lyhrs`, `expr`, `exprsq`, and `hospwork`, respectively. Some IPW estimates given below use an indicator variable for self-employment (`selfempl`) as an auxiliary variable. A much more complete set of regressors for these data is used in the panel attrition study by [Cheng and Trivedi \(2015\)](#).

The data are stored as one line per (individual, wave) pair. If an individual is not interviewed in a given wave, then no observation appears for that wave. Note that an alternative way that the dataset could be configured is to include an observation even when there is no interview. Then a noninterview observation will provide data on key survey-specific variables such as the individual identifier and the wave, but all interview-specific variables would be set to missing.

The extent of attrition

We first drop any (individual, wave) pair for which an interview occurred but data are missing on variables used in this analysis. The `xtdescribe` command is then used to obtain the patterns of missing data due to noninterview.

```

. * Pattern of missing observations in panel dataset
. qui use mus219mabelunbalsmall, clear
. drop if lyearn==.|lyhrs==.|expr==.|exprsq==.|hospwork==.|selfempl==.
(5 observations deleted)
. xtset
Panel variable: id (unbalanced)
Time variable: wave, 1 to 3, but with gaps
    Delta: 1 unit
. xtdescribe
    id: 1100001, 1100003, ..., 3100399                                n =      1933
    wave: 1, 2, ..., 3                                              T =        3
        Delta(wave) = 1 unit
        Span(wave) = 3 periods
        (id*wave uniquely identifies each observation)

Distribution of T_i:   min      5%     25%     50%     75%     95%     max
                      1       1       1       2       3       3       3
    Freq.  Percent   Cum. |  Pattern
    732    37.87  37.87 |  111
    338    17.49  55.35 |  1..
    242    12.52  67.87 |  11.
    204    10.55  78.43 |  ..1
    166     8.59  87.02 |  .11
    164     8.48  95.50 |  .1.
    87      4.50 100.00 |  1.1
    1933   100.00 |  XXX

```

Many individuals have missing data for at least one of the three waves. In some cases, an individual with data missing in one wave reappears in the next wave. And in some cases, individuals enter the sample after the first wave because the survey included refreshment sampling. In the survey, 732 individuals have data available for all three waves, 495 ($= 242 + 166 + 87$) have data for 2 waves, and 706 ($= 338 + 204 + 164$) have data for 1 wave. In total, there are 3,892 ($= 3 \times 732 + 2 \times 495 + 1 \times 706$) observations on 1,933 individuals.

The following gives summary statistics on the key variables.

```
. * Summarize the data
. summarize, sep(0)
```

Variable	Obs	Mean	Std. dev.	Min	Max
id	3,892	1263677	454508.2	1100001	3100399
wave	3,892	1.946043	.8137644	1	3
yearn	3,892	134626.3	88295.98	1000	1002000
yhrs	3,892	1684.091	711.9675	52	5200
female	3,892	1	0	1	1
expr	3,892	18.43358	9.836534	0	58
hospwork	3,892	.1682939	.3741752	0	1
selfempl	3,892	.2546249	.4357061	0	1
lyearn	3,892	11.62464	.6317774	6.907755	13.81751
lyhrs	3,892	7.317732	.5202118	3.951244	8.556414
exprsq	3,892	436.5295	403.556	0	3364

We create a set of selection variables `s1-s3` that indicate for each wave for which an individual appears in the dataset which of the three waves he or she appears in. For example, for an individual with missing pattern 1.1, we have `s1=1`, `s2=0`, and `s3=1` in each of `wave==1` and `wave==3`, while there is no observation in the dataset for `wave==2`.

```
. * Selection indicators for each(id,wave) indicate which waves are available
. generate s1 = 0
. qui replace s1 = 1 if wave==1
. qui replace s1 = 1 if (wave==2 & !missing(L.lyearn))
. qui replace s1 = 1 if (wave==3 & !missing(L2.lyearn))
. generate s2 = 0
. qui replace s2 = 1 if wave==2
. qui replace s2 = 1 if (wave==1 & !missing(F.lyearn))
. qui replace s2 = 1 if (wave==3 & !missing(L.lyearn))
. generate s3 = 0
. qui replace s3 = 1 if wave==3
. qui replace s3 = 1 if (wave==1 & !missing(F2.lyearn))
. qui replace s3 = 1 if (wave==2 & !missing(F.lyearn))
. generate balanced = (s1==1 & s2==1 & s3==1)
. save mus219mabelfinal, replace
(file mus219mabelfinal.dta not found)
file mus219mabelfinal.dta saved
```

`mus219mabelfinal.dta` with these indicators is saved for subsequent analysis. It includes the indicator `balanced`, which equals one for individuals with data available in all three waves.

As an example, the following tabulates counts of `s2` for each of the three waves.

```
. * Count of how many wave 2 also available in wave 1 and wave 3
. tabulate s2 wave
```

s2	Survey wave			Total
	1	2	3	
0	425	0	291	716
1	974	1,304	898	3,176
Total	1,399	1,304	1,189	3,892

In the survey, 974 of the 1,399 present in wave 1 were also present in wave 2, and 898 of the 1,189 present in wave 3 were also present in wave 2.

For those present at wave 1, we compare the wave 1 earnings for those who were not present in wave 3 (`s3==1`) with those who were still present in wave 3 (`s3==0`). This compares 580 individuals with missing pattern 1.. or 11. to 819 individuals with pattern 111 or pattern 1.1.

```
. * Compare wave 1 earnings by wave 3 attrition status
. capture drop attrit*
. twoway (kdensity lyearn if (wave==1 & s3==1), clstyle(p1))
>      (kdensity lyearn if (wave==1 & s3==0), clstyle(p2)),
>      title("Wave 1 earnings by wave 3 attrition status")
>      legend(label(1 "No attrition") label(2 "Attrition"))
>      legend(pos(10) ring(0) col(1))
```

The first panel of figure 19.1 shows little difference in wave 1 earnings by attrition status in wave 3.

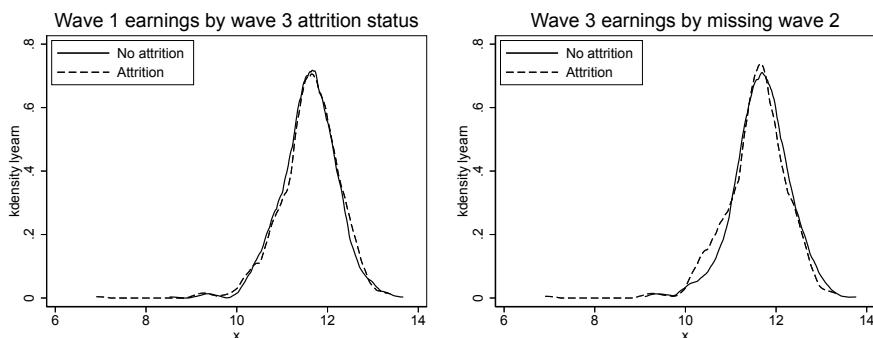


Figure 19.1. Log earnings by attrition status for wave 1 and for wave 3

We next compare, for those in the original wave 1 sample, the wave 3 earnings for the 732 individuals in wave 2 with the 87 individuals who missed wave 2.

```

. * Compare wave 3 earnings by whether missed wave 2
. generate wave2skip = .
(3,892 missing values generated)
. qui replace wave2skip = 1 if (s1==1 & s2==0)
. qui replace wave2skip = 0 if (s1==1 & s2==1)
. twoway (kdensity lyearn if (wave==3 & wave2skip==0), clstyle(p1))
>     (kdensity lyearn if (wave==1 & wave2skip==1), clstyle(p2)),
>     title("Wave 3 earnings by missing wave 2")
>     legend(label(1 "No attrition") label(2 "Attrition"))
>     legend(pos(10) ring(0) col(1))

```

The second panel of figure 19.1 shows that the wave 3 log-earnings distribution for those who skipped wave 2 appears to be around 0.10 to the left of that for those who did not, corresponding to a 10% lower level of earnings.

We perform a difference in means test.

```
. * Difference in means test of wave 3 earnings by whether missed wave 2  
. ttest lyearn if wave==3, by(wave2skip) unequal  
Two-sample t test with unequal variances
```

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf. interval]	
0	732	11.67776	.0229297	.6203751	11.63274	11.72278
1	87	11.53076	.0693067	.6464502	11.39299	11.66854
Combined	819	11.66215	.0218196	.6244371	11.61932	11.70497
diff		.1469954	.0730013		.0022585	.2917322

```

diff = mean(0) - mean(1)                                t = 2.0136
H0: diff = 0                                         Satterthwaite's degrees of freedom = 105.708
Ha: diff < 0                                         Ha: diff != 0                               Ha: diff > 0
Pr(T < t) = 0.9767                                     Pr(|T| > |t|) = 0.0466                  Pr(T > t) = 0.0233

```

The difference in means of 0.147 is statistically significant at level 0.05. The command `regress lyearn wave3skip if wave==3, vce(robust)` leads to equivalent results, aside from slight difference in degrees-of-freedom correction.

19.11.2 Unbalanced and balanced panel analysis

The most common procedure when there is panel attrition is to assume that attrition does not affect the consistency of the usual estimators. This requires the exogenous sampling assumption that data are MAR conditional on regressors. In the case of pooled OLS, for example, we assume that $E(u_{it}|\mathbf{x}_{it}) = 0$ and that $s_{it} = h(\mathbf{x}_{it})$ for some function $h(\cdot)$.

In this situation, we can use either the original unbalanced sample or the balanced sample. The unbalanced sample has the advantage of having more observations. The balanced sample can have the advantage that some estimation methods may apply only to balanced samples, though the basic panel commands can be implemented on either unbalanced or balanced samples.

Creating a balanced dataset

One way to create a balanced dataset is to drop any individual who is not observed in all three waves. We earlier defined variable `balanced = (s1==1 & s2==1 & s3==1)`.

```

. * Manually create balanced dataset
. keep if balanced == 1
(1,696 observations deleted)

. xtdescribe
      id: 1100001, 1100003, ..., 1103899          n =      732
      wave: 1, 2, ..., 3                          T =       3
          Delta(wave) = 1 unit
          Span(wave) = 3 periods
          (id*wave uniquely identifies each observation)

Distribution of T_i:    min      5%     25%     50%     75%     95%     max
                      3        3        3        3        3        3        3
      Freq.  Percent   Cum. |  Pattern
      732    100.00 100.00 |  111
      732    100.00           |  XXX

```

Of the original 3,892 observations, 1,696 were dropped, leaving 2,196 (= 3892 – 1696) observations on 732 (= 2196/3) individuals.

An alternative is to use the community-contributed `xtbalance` command ([Yujun 2009](#)). This command has a required option `range()` that gives the range of the time identifier to be used. Here we want to use `wave` equal to 1, 2, and 3, so we use option `range(1 3)`. Again reading in the original dataset, we use the `miss()` option to additionally drop observations that are missing data on those variables to be used in the analysis below.

```

. * xtbalance command creates a balanced dataset
. qui use mus219mabelunbalsmall, clear
. global xlist lyhrs expr exprsq hospwork
. xtbalance, range(1 3) miss(lyearn $xlist)
(5 observations deleted due to missing)
(1696 observations deleted due to discontinues)

```

```

. xtdescribe
      id: 1100001, 1100003, ..., 1103899          n =      732
      wave: 1, 2, ..., 3                          T =       3
          Delta(wave) = 1 unit
          Span(wave) = 3 periods
          (id*wave uniquely identifies each observation)

Distribution of T_i:   min      5%     25%     50%     75%     95%     max
                           3        3        3        3        3        3        3
      Freq.  Percent    Cum. |  Pattern
      -----+-----
      732    100.00 100.00 |  111
      -----+-----
      732    100.00           |  XXX

```

OLS, random-effects, and fixed-effects estimates

We obtain OLS, random-effects (RE), and fixed-effects (FE) estimators for unbalanced and balanced samples.

```

. * OLS, RE, and FE regressions for unbalanced sample
. * and for balanced sample
. qui use mus219mabelfinal, clear // Unbalanced sample
. qui regress lyearn $xlist, vce(cluster id)
. estimates store OLS_Unbal
. qui xtreg lyearn $xlist, re vce(robust)
. estimates store RE_Unbal
. qui xtreg lyearn $xlist, fe vce(robust)
. estimates store FE_Unbal
. qui regress lyearn $xlist if balanced==1, vce(cluster id) // Balanced sample
. estimates store OLS_Bal
. qui xtreg lyearn $xlist if balanced==1, re vce(robust)
. estimates store RE_Bal
. qui xtreg lyearn $xlist if balanced==1, fe vce(robust)
. estimates store FE_Bal

```

The following table compares the results.

```

. * Table comparing OLS, RE, and FE results for balanced
. * and unbalanced data
. estimates table OLS_Unbal OLS_Bal RE_Unbal RE_Bal FE_Unbal FE_Bal,
> b(%7.3f) se stats(N r2)

```

Variable	OLS_U^1	OLS_Bal	RE_Un^1	RE_Bal	FE_Un^1	FE_Bal
lyhrs	0.758	0.832	0.609	0.617	0.294	0.305
	0.027	0.037	0.031	0.041	0.040	0.052
expr	0.013	-0.003	0.016	-0.000	-0.142	-0.133
	0.003	0.005	0.003	0.006	0.033	0.039
exprsq	-0.000	0.000	-0.000	-0.000	0.002	0.002
	0.000	0.000	0.000	0.000	0.001	0.001
hospwork	0.044	0.063	0.023	0.058	-0.006	0.028
	0.027	0.040	0.024	0.033	0.038	0.042
_cons	5.958	5.598	7.018	7.132	11.116	11.152
	0.198	0.261	0.225	0.296	0.401	0.546
N	3892	2196	3892	2196	3892	2196
r2	0.405	0.435			0.079	0.081

Legend: b/se

Despite the large difference in the number of observations in unbalanced versus balanced datasets (3,892 versus 2,196), for a given estimator, there is little difference in the coefficient of the highly statistically significant regressor `lyhrs`. Much more pronounced is the difference across the OLS, RE, and FE estimates. As expected, the unbalanced estimates are more precise because they are based on more observations.

First-difference estimates

The individual-specific FE model has the attraction that additionally conditioning on the fixed effect makes the conditional MAR assumption more reasonable.

Given $y_{it} = \alpha_i + \mathbf{x}'_{it}\beta + u_{it}$, we assume that $E(u_{it}|\alpha_i, \mathbf{x}_{it}) = 0$ and that $s_{it} = h(\mathbf{x}_{it})$ for some function $h(\cdot)$. So we assume that after controlling for a time invariant individual-specific fixed effect, attrition depends only on the included regressors. Fixed-effects or first-difference estimators then eliminate these individual-specific fixed effects.

The first-difference estimates are obtained as follows:

```

. * Compare first-difference results for balanced and unbalanced data
. global FDxlist D.lyhrs D.expr D.exprsq D.hospwork
. sort id wave
. qui regress D.lyearn $FDxlist, cluster(id)
. estimates store FD_Unbal
. qui regress D.lyearn $FDxlist if balanced==1, cluster(id) // Balanced sample
. estimates store FD_Bal

. estimates table FD_Unbal FD_Bal, b(%7.4f) se stats(N r2)

```

Variable	FD_Un~1	FD_Bal
lyhrs		
D1.	0.2766 0.0440	0.3034 0.0562
expr		
D1.	-0.0970 0.0372	-0.1142 0.0455
exprsq		
D1.	0.0021 0.0008	0.0020 0.0010
hospwork		
D1.	0.0051 0.0421	0.0316 0.0453
_cons	0.0321 0.0135	0.0171 0.0161
N	1872	1464
r2	0.0600	0.0676

Legend: b/se

The results are similar across the two samples and are similar to the preceding FE estimates. For the larger unbalanced dataset, the elasticity of annual earnings with respect to hours is 0.277 and is very precisely estimated with $t = 6.29$. The experience variables are jointly statistically significant at level 0.05.

19.11.3 Inverse-probability weighting

We now suppose that conditioning on regressors included in the model is insufficient to control for bias induced by attrition.

The IPW method overcomes this by adding additional variables as controls for selection. For brevity, we illustrate the IPW method for OLS estimation in first differences. Analysis for OLS estimation in levels, possible when it is reasonable to use a model without individual-specific effects, is qualitatively similar.

Given $y_{it} = \alpha_i + \mathbf{x}'_{it}\beta + \mathbf{x}'_{2i}\delta + u_{it}$, where \mathbf{x}_{it} are time-varying variables and \mathbf{x}_{2i} are time-invariant variables, first differencing yields

$$\Delta y_{it} = \Delta \mathbf{x}'_{it}\beta + \Delta u_{it}$$

Note that the time-invariant variables have disappeared.

Let the selection indicator $s_{it} = 1$ if data on individual i are available at time t and $s_{it} = 0$ otherwise. Then data for the first-difference regression in time t are available if $s_{it} = 1$ and $s_{i,t-1} = 1$. The simplest approach is to assume that attrition in any period t depends only on base period variables \mathbf{z}_{i1} ; later analysis relaxes this specification. We assume that these variables alone explain any attrition, so that

$$\Pr(s_{it} = 1 | \mathbf{z}_{i1}, s_{i,t-1} = 1, \Delta y_{it}, \Delta \mathbf{x}_{it}) = \Pr(s_{it} = 1 | \mathbf{z}_{i1}) = \Phi(\mathbf{z}'_{i1}\gamma_t) \quad (19.9)$$

We specify a probit model because the sample-selection model presented later uses the probit. The logit model is more commonly used for IPW; in practice, the weights obtained using either logit or probit will be similar.

The key decision is what variables to include in \mathbf{z}_{i1} . It can include at least \mathbf{x}_{it} at $t = 1$ and the time-invariant regressors (\mathbf{x}_{2i}) that were first-differenced out.

Given estimates $\hat{\gamma}$ of the probit model, the IPW estimator solves the weighted LS first-order conditions

$$\sum_{i=1}^N \frac{s_{it}}{\Phi(\mathbf{z}'_{i1}\hat{\boldsymbol{\gamma}}_t)} \Delta\mathbf{x}_{it} (\Delta y_{it} - \Delta\mathbf{x}'_{it}\boldsymbol{\beta}) = \mathbf{0}$$

This can be implemented in Stata using the `regress` command with dependent variable Δy_{it} , regressors $\Delta\mathbf{x}_{it}$, and command qualifier [`pweight=weight`], where `weight` equals $s_{it}/\Phi(\mathbf{z}'_{i1}\hat{\boldsymbol{\gamma}})$. This weight equals 0 for missing observations and $1/\Phi(\mathbf{z}'_{i1}\hat{\boldsymbol{\gamma}})$ for nonmissing observations.

Filled-in panel dataset

The current dataset includes only observations with $s_{it} = 1$. The subsequent analysis requires a balanced dataset with all three waves per individual; in waves where the individual was not interviewed, all variables will then be set to missing.

To do so, we use the `fillin` command, which adds observations with missing data so that the dataset includes all interactions of the variables provided as arguments of the command, here `id` and `wave`. We include a listing of some observations to make clear that the observations added have missing values for all variables but `id` and `wave`. We also redefine the selection indicators `s1-s3`.

```
. * Expand dataset to include missing waves 2 and 3 given present in wave 1
. qui use mus219mabelfinal, clear
. fillin id wave
. sort id wave
```

```

. list id wave s1 s2 s3 lyearn _fillin in 13/24, clean
      id    wave    s1    s2    s3    lyearn    _fillin
13. 1100014      1    1    1    10.81978    0
14. 1100014      2    1    1    10.59663    0
15. 1100014      3    1    1    10.50507    0
16. 1100015      1    1    0    0    11.60824    0
17. 1100015      2    .    .    .    .    1
18. 1100015      3    .    .    .    .    1
19. 1100016      1    1    0    0    12.25486    0
20. 1100016      2    .    .    .    .    1
21. 1100016      3    .    .    .    .    1
22. 1100018      1    1    1    0    11.28978    0
23. 1100018      2    1    1    0    11.28978    0
24. 1100018      3    .    .    .    .    1

. generate swave = s1
(1,907 missing values generated)
. replace swave = . if s1==0
(700 real changes made, 700 to missing)
. drop s1 s2 s3
. generate s1 = 0
. qui replace s1 = 1 if (wave==1 & !missing(lyearn))
. qui replace s1 = 1 if (wave==2 & !missing(L.lyearn))
. qui replace s1 = 1 if (wave==3 & !missing(L2.lyearn))
. generate s2 = 0
. qui replace s2 = 1 if (wave==2 & !missing(lyearn))
. qui replace s2 = 1 if (wave==1 & !missing(F.lyearn))
. qui replace s2 = 1 if (wave==3 & !missing(L.lyearn))
. generate s3 = 0
. qui replace s3 = 1 if (wave==3 & !missing(lyearn))
. qui replace s3 = 1 if (wave==1 & !missing(F2.lyearn))
. qui replace s3 = 1 if (wave==2 & !missing(F.lyearn))
. sum id wave s1 s2 s3 lyearn swave, sep(0)

```

Variable	Obs	Mean	Std. dev.	Min	Max
id	5,799	1373326	585159.2	1100001	3100399
wave	5,799	2	.816567	1	3
s1	5,799	.7237455	.4471828	0	1
s2	5,799	.6745991	.4685649	0	1
s3	5,799	.6151061	.4866122	0	1
lyearn	3,892	11.62464	.6317774	6.907755	13.81751
swave	3,192	1	0	1	1

The added observations have missing values for all variables aside from `id` and `wave`. The dataset is now a square dataset with 5,799 observations on 1,933 individuals ($= 5799/3$), but nonmissing data are available for only 3,892 observations. The `s3` variable, for example, indicates that there are $0.6151 \times 5799/3 = 1189$ observations with nonmissing wave-3 data.

IPW estimates with selection based on wave-1 data

We first consider estimation of the first-difference equation for log-earnings using only wave-2 data.

To control for attrition, we fit a probit selection model for $s_{i2} = 1$, where we use as \mathbf{z}_{i1} the value of the level of the log-earnings equation variables in wave 1. We have

```

. * IPW: (1) Probit selection equation for wave 2 given in wave 1
. global IPWxlist L.lyhrs L.expr L.exprsq L.hospwork
. probit s2 $IPWxlist if wave==2
Iteration 0: log likelihood = -859.04215
Iteration 1: log likelihood = -852.88607
Iteration 2: log likelihood = -852.88502
Iteration 3: log likelihood = -852.88502
Probit regression
Number of obs = 1,399
LR chi2(4) = 12.31
Prob > chi2 = 0.0152
Pseudo R2 = 0.0072
Log likelihood = -852.88502

```

s2	Coefficient	Std. err.	z	P> z	[95% conf. interval]
lyhrs L1.	.0276565	.067977	0.41	0.684	-.1055759 .160889
expr L1.	.0356527	.0133619	2.67	0.008	.0094638 .0618416
exprsq L1.	-.0006163	.0003051	-2.02	0.043	-.0012143 -.0000183
hospwork L1.	-.0429673	.0960163	-0.45	0.655	-.2311558 .1452212
_cons	-.0872095	.49163	-0.18	0.859	-1.050787 .8763676

```

. qui predict s2prob if e(sample)==1
. tabulate s2 if e(sample)==1

```

s2	Freq.	Percent	Cum.
0	425	30.38	30.38
1	974	69.62	100.00
Total		1,399	100.00

```
. tabstat s2prob if e(sample)==1, stats(count min p10 p50 p90 max)
```

Variable	N	Min	p10	p50	p90	Max
s2prob	1399	.5250311	.6233922	.714309	.7368833	.7437167

This estimate is based on 1,399 individuals, 974 individuals observed in both waves 1 and 2 (so wave-2 first-differences data are available), and 425 present in wave 1 but not in wave 2. The predicted probabilities range from 0.525 to 0.744. The probability of continuing to the wave-2 survey at first increases with years of experience and then decreases after 29 years [$= 0.0356/(2 \times 0.000616)$].

We then obtain the IPW estimates by weighted OLS regression for the 974 individuals observed in both waves 1 and 2.

```
. * IPW: (2) Generate weights and do weighted and unweighted first-difference OLS
. qui generate s2weight = s2/s2prob if wave==2
. tabstat s2weight if wave==2 & s2==1, stats(count min p10 p50 p90 max)
```

Variable	N	Min	p10	p50	p90	Max
s2weight	974	1.344598	1.356602	1.397984	1.590194	1.824038

```
. regress D.lyearn $FDxlist [pweight=s2weight] if wave==2, vce(robust) // Weighted
(sum of wgt is 1,398.77602756023)
```

Linear regression	Number of obs	=	974
	F(4, 969)	=	11.33
	Prob > F	=	0.0000
	R-squared	=	0.0764
	Root MSE	=	.43776

D.lyearn	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]
lyhrs D1.	.3036735	.0499019	6.09	0.000	.2057453 .4016017
expr D1.	-.079066	.0930013	-0.85	0.395	-.2615731 .1034412
exprsq D1.	.0022219	.0008784	2.53	0.012	.0004981 .0039457
hoswork D1.	.0528654	.0589415	0.90	0.370	-.0628022 .168533
_cons	.0617818	.0846183	0.73	0.465	-.1042745 .227838

```
. estimates store IPW2
```

The weights range from 1.345 to 1.824. Observations are multiplied by the square root of these weights, so there is at most a 16% difference (because $\sqrt{1.824/1.345} = 1.16$) in the weights on any single observation, a modest

difference. The estimates are similar to the unweighted first-difference OLS estimates obtained previously.

The following code repeats the exercise for OLS estimation of the first-difference equation using only wave-3 data. We fit a probit selection model for $s_{i3} = 1$, where we again use as \mathbf{z}_{i1} the value of variables in wave 1. It is important that the probit model estimation be restricted to respondents with data also available in wave 2 because subsequent weighted first-difference LS estimation in wave 3 requires wave-2 data. We have

```
. * IPW: Now do wave 3 with selection on wave 1 variables
. global IPWxlist3 L2.lyhrs L2.expr L2.exprsq L2.hospwork
. qui probit s3 $IPWxlist3 if wave==3 & s2==1
. qui predict s3prob if e(sample)
. tabulate s3 if e(sample)==1



| s3    | Freq. | Percent | Cum.   |
|-------|-------|---------|--------|
| 0     | 242   | 24.85   | 24.85  |
| 1     | 732   | 75.15   | 100.00 |
| Total |       | 974     | 100.00 |



. qui generate s3weight = s3/s3prob if wave==3
. tabstat s3prob s3weight if e(sample)==1, stats(count min p10 p50 p90 max) col(s)



| Variable | N   | Min      | p10      | p50      | p90      | Max      |
|----------|-----|----------|----------|----------|----------|----------|
| s3prob   | 974 | .4396037 | .6424823 | .7758189 | .8131418 | .9041119 |
| s3weight | 974 | 0        | 0        | 1.259265 | 1.454074 | 2.100374 |



. qui regress D.lyearn $FDxlist [pweight=s3weight] if wave==3, vce(robust)
. estimates store IPW3
```

The probit estimates are based on 974 individuals, 732 observed in waves 2 and 3 and 242 observed in wave 2 but not wave 3. The final weighted estimates use the 732 observed in waves 2 and 3. The estimates will be presented in a table below.

We then combine the two waves and do pooled weighted LS estimation of the first-difference equation using waves 2 and 3, using the previously obtained weights. We have

```

. * IPW: Estimate waves 2 and waves 3 together
. qui generate sweight = .
. qui replace sweight = s2weight if wave==2
. qui replace sweight = s3weight if wave==3
. tabstat sweight if sweight!=0, stats(count min p10 p50 p90 max) col(s)
Variable | N      Min      p10      p50      p90      Max
sweight | 1706  1.106058  1.248824  1.372499  1.555892  2.100374

```

```

. qui regress D.lyearn $FDxlist [pweight=sweight], vce(robust)
. estimates store IPWboth

```

The estimates will be presented in a table below.

IPW estimates with selection based on time-varying data

The previous analysis uses only data available at wave 1 to estimate the selection probabilities. We could instead use more recent data as it becomes available.

Thus, in the probit selection equation for wave 3, we could use wave-2 data. In that case, subsequent OLS estimation should use a weight of $s_{i3}/\{\Phi(\mathbf{z}'_{i1}\hat{\gamma}_2) \times \Phi(\mathbf{z}'_{i1}\hat{\gamma}_3)\}$. This requires only a minor change to the code given previously. It does require stronger assumptions to ensure that the IPW estimator is consistent, however; see [Wooldridge \(2010, 842\)](#).

IPW based on auxiliary regressors

A separate issue is that the IPW estimator places high weight on observations with the probability that $s_{it} = 1$ predicted to be close to zero. This can potentially generate instability in the weighted regressions.

An alternative IPW estimator reduces the likelihood of this by estimating two different binary outcome models. One model leads to predicted selection probability $\Phi(\mathbf{z}'_{i1}\hat{\gamma})$, as before. A second model adds auxiliary variables \mathbf{v}_i , say, leading to selection probability $\Phi(\mathbf{z}'_{i1}\tilde{\gamma} + \mathbf{v}'_{i1}\tilde{\delta})$. The IPW estimator then uses weight $s_{it} \times \Phi(\mathbf{z}'_{i1}\hat{\gamma})/\Phi(\mathbf{z}'_{i1}\tilde{\gamma} + \mathbf{v}'_{i1}\tilde{\delta})$.

The following example implements this alternative weighting for wave-2 first-difference OLS. We add `selfempl`, an indicator variable equal to one if self-employed, as an auxiliary variable. Then,

```
. * IPW: Alternative weights based on auxiliary variables wave 2 versus wave 1
. qui probit s2 $IPWxlist L.selfempl if wave==2
. qui predict s2probaugment if e(sample)
. qui generate s2relweight=s2prob/s2probaugment
. tabstat s2relweight if e(sample)==1, stats(count min p10 p50 p90 max) col(s)



| Variable    | N    | Min      | p10     | p50      | p90      | Max      |
|-------------|------|----------|---------|----------|----------|----------|
| s2relweight | 1399 | .9745781 | .980992 | .9900049 | 1.031482 | 1.074542 |

. sum s2prob s2probaugment s2relweight


| Variable     | Obs   | Mean     | Std. dev. | Min      | Max      |
|--------------|-------|----------|-----------|----------|----------|
| s2prob       | 1,399 | .6961762 | .0435256  | .5250311 | .7437167 |
| s2probaugm^t | 1,399 | .6961827 | .0460661  | .4950356 | .7599604 |
| s2relweight  | 1,399 | 1.000462 | .021652   | .9745781 | 1.074542 |


. qui regress D.lyearn $FDxlist [pweight=s2relweight] if wave==2, vce(robust)
. qui estimates store IPW2Aug
```

Comparison of IPW estimates

The following table compares unweighted first-differences OLS estimates using wave-2 and 3 data with the various IPW estimates.

```

. * Unweighted, IPW, and sample-selection estimates: First-difference in wave 2
. qui regress D.lyearn $FDxlist if wave==2, vce(robust) noheader // Unweighted
. estimates store FD_UNW
. estimates table FD_UNW IPW2 IPW3 IPWboth IPW2Aug, b(%7.4f) eq(1) se stats(N r2)

```

Variable	FD_UNW	IPW2	IPW3	IPWboth	IPW2Aug
lyhrs					
D1.	0.3034 0.0505	0.3037 0.0499	0.3083 0.0776	0.3054 0.0424	0.3019 0.0504
expr					
D1.	-0.0677 0.0913	-0.0791 0.0930	0.0121 0.1496	-0.1148 0.0394	-0.0713 0.0924
exprsq					
D1.	0.0022 0.0009	0.0022 0.0009	0.0000 0.0033	0.0021 0.0008	0.0022 0.0009
hospwork					
D1.	0.0598 0.0601	0.0529 0.0589	-0.0235 0.0526	0.0198 0.0406	0.0591 0.0598
_cons	0.0710 0.0829	0.0618 0.0846	0.0161 0.0164	0.0185 0.0162	0.0677 0.0840
N	974	974	732	1706	974
r2	0.0737	0.0764	0.0643	0.0722	0.0728

Legend: b/se

There is little difference in the highly statistically significant regressor Δ_{lyhrs} across the various models.

19.11.4 Selection-based panel attrition model

We continue with the example of first-difference OLS with attrition. The IPW estimator can provide consistent estimates in situations where balanced or unbalanced estimation is inconsistent. It requires the conditional MAR or selection-on-observables assumption (19.9); that is, that

$$\Pr(s_{it} = 1 | \mathbf{z}_{i1}, s_{i,t-1} = 1, \Delta y_{it}, \Delta \mathbf{x}_{it}) = \Pr(s_{it} = 1 | \mathbf{z}_{i1}).$$

If this assumption fails, then we are in a setting of MNAR or selection on unobservables. Then we use a two-equation system for Δy_{it} and s_{it} with errors that are correlated across equations. This model is a panel extension of the selection model of section 19.6.

We first consider a model without fixed effects. Define latent variables for the outcome (y) and selection indicator (s),

$$\begin{aligned} y_{it}^* &= \mathbf{x}'_{it}\boldsymbol{\beta} + \mathbf{x}'_{2i}\boldsymbol{\beta}_2 + \varepsilon_{1it} \\ s_{it}^* &= \mathbf{z}'_{it}\boldsymbol{\gamma} + \varepsilon_{2it} \end{aligned}$$

where the equation errors $(\varepsilon_{1it}, \varepsilon_{2it})$ may be correlated. In the two-component vector $(\mathbf{x}_{it} \mathbf{x}_{2i})$, the first component \mathbf{x}_{it} consists of time-varying regressors, and the second component \mathbf{x}_{2i} consists of time-invariant regressors. We observe

$$\begin{aligned} y_{it} &= s_{it} \times y_{it}^* \\ s_{it} &= \mathbf{1}(s_{it}^* > 0) \end{aligned}$$

In this selection model, the random shock, ε_2 , which affects the probability of attrition is potentially correlated with the shock ε_1 , which affects the outcome. Ignoring this correlation, as when the outcome equation is estimated under conditional MAR assumptions, results in selection bias.

A number of panel-data estimators are available for estimating the selection model; see [Wooldridge \(2010\)](#), chap. 19.9). As in the case of the classic selection model for cross-sectional data, robust identification of the outcome parameter $\boldsymbol{\beta}$ requires that the attrition equation contain some nontrivial regressors that do not directly affect the outcome. One potential difference from the cross-sectional case, however, comes from the possibility that these variables can vary over t . Furthermore, with attrition, not only y_{it} is missing when $s_{it} = 1$ but also \mathbf{x}_{it} .

First-difference model

Rather than illustrate sample-selection methods for the preceding model in levels, we analyze the qualitatively similar first-difference model that has the advantage of being robust to the presence of time-invariant individual-specific effects.

The selection model equations are then

$$\begin{aligned}\Delta y_{it}^* &= \Delta \mathbf{x}'_{it} \boldsymbol{\beta} + \Delta \varepsilon_{1it} \\ s_{it}^* &= \mathbf{z}'_{it} \boldsymbol{\gamma} + \varepsilon_{2it}\end{aligned}$$

where the equation errors $(\Delta \varepsilon_{1it}, \varepsilon_{2it})$ may be correlated. We observe

$$\begin{aligned}\Delta y_{it} &= s_{it} \times \Delta y_{it}^* \\ s_{it} &= \mathbf{1}(s_{it}^* > 0)\end{aligned}$$

We first consider the selection probability. Assume that, conditional on $s_{i,t-1} = 1$ and \mathbf{z}_{it} , the selection error $\varepsilon_{2it} \sim N(0, 1)$. Then,

$$\Pr(s_{it} = 1 | s_{i,t-1} = 1, \mathbf{z}_{it}) = \Phi(\mathbf{z}'_{it} \boldsymbol{\gamma})$$

a probit model.

Second, assume that the error in the outcome equation is a multiple of the selection error plus independent noise, so

$$\Delta \varepsilon_{1it} = \rho \varepsilon_{2it} + \eta_{it}$$

where η_{it} is an independent error and we again conditional on $s_{i,t-1} = 1$. By previous results in section [19.6](#), it follows that

$$E(\Delta y_{it} | s_{i,t-1} = 1) = \Delta \mathbf{x}'_{it} \boldsymbol{\beta} + \rho \lambda(\mathbf{z}'_{it} \boldsymbol{\gamma})$$

where $\lambda(z) = \phi(z)/\Phi(z)$ is the inverse Mills ratio.

A consistent estimator of $\lambda(\mathbf{z}'_{it} \boldsymbol{\gamma})$, denoted $\widehat{\lambda}_{it}$, is generated by the probit equation for the attrition event. Then, estimate by pooled OLS using only the nonmissing observations the equation

$$\Delta y_{it} = \Delta \mathbf{x}'_{it} \boldsymbol{\beta} + \rho \widehat{\lambda}_{it} + \left\{ \eta_{it} + \Delta \varepsilon_{1it} + \rho_t (\lambda_{it} - \widehat{\lambda}_{it}) \right\}$$

where the three terms inside the curly brackets define the composite error on the outcome equation. Under the assumption that all elements of $\Delta\mathbf{x}_{it}$ are uncorrelated with the composite error term, the LS estimator is a consistent estimator. However, because $\widehat{\lambda}_{it}$ is a generated regressor, inference should be based on, for example, a cluster bootstrap that is a cluster variant of the two-step bootstrap given in section 12.4.5.

A test of the null hypothesis of conditional MAR against the alternative of selection bias may be based on $H_0 : \rho = 0$ versus $H_1 : \rho \neq 0$. Given quite strong assumptions involved in its implementation and the complexity of the robust variance estimator, the outcome of the test should be treated with caution.

More generally, we may estimate the selection equation separately in each time period. So assume that $\Pr(s_{it} = 1 | s_{i,t-1} = 1, \mathbf{z}_{it}) = \Phi(\mathbf{z}'_{it}\boldsymbol{\gamma}_t)$ and $\Delta\varepsilon_{1it} = \rho_t \varepsilon_{2it} + \eta_{it}$, which in turn generates the time-varying Mills ratio term $\lambda(\mathbf{z}'_{it}\boldsymbol{\gamma}_t)$. After obtaining $\widehat{\lambda}(\mathbf{z}'_{it}\boldsymbol{\gamma}_t)$ by separate probit estimation for each time period, we then do pooled OLS estimation for all periods of the model

$$\Delta y_{it} = \Delta\mathbf{x}'_{it}\boldsymbol{\beta} + \rho_2 d_{2it} \widehat{\lambda}_{it} + \cdots + \rho_T d_{Tit} \widehat{\lambda}_{it} + v_{it}$$

where $d_{j_{it}} = 1$ if $t = j$ and $d_{j_{it}} = 0$ otherwise. Again, inference should be based on a cluster bootstrap for a two-step estimator.

An alternative specification is that in which one or more elements of \mathbf{x}_{it} are endogenous. Then an IV or GMM-type estimator would be preferred. The usual caveats regarding the choice of instruments will apply, and note that the presence of serially correlated errors will affect both the selection of valid instruments and the appropriate variance estimator.

Selection model estimates

Here we present the MLE for the pooled data, although there are reasons to prefer the two-step estimator. Two-step estimation is left as an exercise for the interested reader.

We use as the variables \mathbf{z}_{it} in the selection equation the lagged-level values of the variables in the outcome equation, as well as the lagged value of the self-employment indicator variable `selfempl`.

We first fit a pooled version of the Heckman selection model, jointly estimating both the selection equation and the (first-differenced) outcome equation over waves 2 and 3. We use ML estimates under the additional assumption of normality because this can provide cluster-robust standard errors.

```
. * Sample-selection model: Pooled first-difference OLS using wave 2 and 3 data
. generate dsel =
(5,799 missing values generated)
. replace dsel = s2 if wave==2
(1,933 real changes made)
. replace dsel = s3 if wave==3
(1,933 real changes made)
. tabulate dsel wave, missing
```

dsel	Survey wave			Total
	1	2	3	
0	0	629	744	1,373
1	0	1,304	1,189	2,493
.	1,933	0	0	1,933
Total	1,933	1,933	1,933	5,799

```

. heckman D.lyearn $FDxlist, select(dsel = $IPWxlist L.selfempl)
>      nolog vce(cluster id)

Heckman selection model                               Number of obs     =      2,703
(regression model with sample selection)           Selected      =      1,872
                                                   Nonselected    =       831
                                                   Wald chi2(4)    =      46.44
Log pseudolikelihood = -2672.479                  Prob > chi2     =     0.0000
                                                   (Std. err. adjusted for 1,729 clusters in id)

```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
D_lyearn	lyhrs					
	D1.	.2774803	.0438924	6.32	0.000	.1914528 .3635078
	expr					
	D1.	-.0919311	.0392842	-2.34	0.019	-.1689266 -.0149355
	exprsq					
	D1.	.001918	.0008987	2.13	0.033	.0001565 .0036795
dsel	hospwk					
	D1.	.0059664	.0422144	0.14	0.888	-.0767723 .088705
	_cons					
		.0107341	.0396295	0.27	0.786	-.0669384 .0884066
	lyhrs					
	L1.	-.0349352	.0586778	-0.60	0.552	-.1499415 .0800711
/athrho	expr					
	L1.	.0563963	.0097657	5.77	0.000	.0372558 .0755367
	exprsq					
	L1.	-.0010023	.0002369	-4.23	0.000	-.0014667 -.0005379
	hospwk					
	L1.	-.0592832	.0709145	-0.84	0.403	-.1982731 .0797067
/lnsigma	selfempl					
	L1.	-.0607765	.0695311	-0.87	0.382	-.1970549 .0755019
	_cons					
		.1795994	.4322929	0.42	0.678	-.6676791 1.026878
	/athrho					
		.1005674	.175148	0.57	0.566	-.2427165 .4438512
rho	/lnsigma					
		-.8637342	.0398811	-21.66	0.000	-.9418997 -.7855686
	rho					
sigma		.1002297	.1733885			-.2380599 .4168316
		.4215849	.0168133			.3898865 .4558604
	lambda					
		.0422553	.0733096			-.1014288 .1859394

```

Wald test of indep. eqns. (rho = 0): chi2(1) = 0.33          Prob > chi2 = 0.5658
. estimates store Heckpool

```

The formal test of independent equations has $p = 0.57$, indicating that there is no selection bias problem. We could just estimate the first-difference equation by OLS.

We next fit the selection model separately for wave 2 and wave 3 and generate separate inverse-Mills ratio terms for each wave.

Wave 2 estimation leads to

```
. * Sample-selection model: First-difference OLS for wave 2 given wave 1
. qui heckman D.lyearn $FDxlist if wave==2, select(s2 = $IPWxlist L.selfempl)
>     mills(mills2) vce(robust)
. display "N = " e(N) " Independence test chi2(1) = " e(chi2_c) " p = " e(p_c)
N = 1399 Independence test chi2(1) = 23.096628 p = 1.541e-06
. estimates store Heck2
```

Estimates for the outcome equation are presented in a table below.

And wave-3 estimation leads to

```
. * Sample-selection model: First-difference OLS for wave 3 given wave 1 and
> wave 2
. qui heckman D.lyearn $FDxlist if wave==3, select(s3 = $IPWxlist L.selfempl)
>     mills(mills3) vce(robust)
. display "N = " e(N) " Independence test chi2(1) = " e(chi2_c) " p = " e(p_c)
N = 1304 Independence test chi2(1) = 2.5440936 p = .11070743
. estimates store Heck3
```

Estimates for the outcome equation are presented in a table below.

Pooled OLS estimation with separate inverse-Mills ratio terms for each wave yields

```
. * Sample-selection model: Pooled first-difference OLS with different lambda for
> waves 2 and 3
. generate millsp = mills2
(4,400 missing values generated)
. replace millsp = mills3 if wave==3
(1,304 real changes made)
```

```
. regress D.lyearn $FDxlist i.wave#c.millsp, vce(cluster id)
Linear regression
Number of obs = 1,872
F(6, 1139) = 8.96
Prob > F = 0.0000
R-squared = 0.0610
Root MSE = .4211
(Std. err. adjusted for 1,140 clusters in id)
```

D.lyearn	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]
lyhrs D1.	.2784121	.0438532	6.35	0.000	.19237 .3644541
expr D1.	-.0509594	.0646665	-0.79	0.431	-.1778382 .0759194
exprsq D1.	.0015953	.0009852	1.62	0.106	-.0003376 .0035283
hosptwork D1.	.0075139	.0422797	0.18	0.859	-.0754408 .0904687
wave#c.millsp 2	.1706688	.1442039	1.18	0.237	-.1122662 .4536039
3	.1057655	.1061797	1.00	0.319	-.1025642 .3140952
_cons	-.0232218	.0560811	-0.41	0.679	-.1332557 .0868121

```
. test 2b.wave#c.millsp 3.wave#c.millsp
( 1) 2b.wave#c.millsp = 0
( 2) 3.wave#c.millsp = 0
      F( 2, 1139) = 0.70
      Prob > F = 0.4948
. estimates store Heckboth
```

The standard errors correct for clustering but do not correct for the first-stage estimation of the inverse-Mills ratio terms.

The unweighted pooled OLS estimates and the various Heckman selection model estimates are given in the following table.

```

. * Unweighted, IPW and sample-selection estimates: First-difference in wave 2
. qui regress D.lyearn $FDxlist, vce(robust) noheader
. estimates store FD_UNW
. estimates table FD_UNW Heckpool Heck2 Heck3 Heckboth, b(%7.4f) eq(1)
>   se stats(N r2) keep($FDxlist)

```

Variable	FD_UNW	Heckp~l	Heck2	Heck3	Heckb~h
lyhrs					
D1.	0.2766 0.0398	0.2775 0.0439	0.3015 0.0499	0.2580 0.0632	0.2784 0.0439
expr					
D1.	-0.0970 0.0379	-0.0919 0.0393	-0.0948 0.0878	0.0373 0.1443	-0.0510 0.0647
exprsq					
D1.	0.0021 0.0008	0.0019 0.0009	0.0031 0.0009	-0.0001 0.0032	0.0016 0.0010
hoswork					
D1.	0.0051 0.0377	0.0060 0.0422	0.0658 0.0617	-0.0433 0.0454	0.0075 0.0423
N	1872	2703	1399	1304	1872
r2	0.0600				0.0610

Legend: b/se

19.11.5 Sample refreshment

In practice, panel surveys sometimes anticipate attrition and add an additional (refreshment) sample group to compensate for the loss of observations. This is another alternative to weighting and model-based methods. Correctly implemented, this approach solves the loss of estimation efficiency due to sample reduction in case the MAR assumption holds. If, however, attrition is selective and not random, then replacement by a random refreshment sample does not solve the problem. Whether sample refreshment reduces or eliminates the effects of sample attrition depends upon the design of the additional survey.

Refreshment samples can provide additional information about the attrition process, allowing for more robust and precise estimation than relying solely on conventional methods. However, in practice, the motivation behind

refreshment sampling may be to obtain additional information on a subpopulation of special interest; see [Cheng and Trivedi \(2015\)](#) for an empirical example.

19.12 Additional resources

For tobit estimation, the relevant entries are [R] **tobit**, [R] **tobit postestimation**, [R] **ivtobit**, and [R] **intreg**. Useful community-contributed commands are `clad` and `tobcm`. Various MES can be computed by using `margins` with several different `predict` options. For tobit panel estimation, the relevant command is [XT] **xttobit**, whose application is covered in chapter [22](#). Normal selection models can be fit using the `heckman` command. For details of fitting nonnormal selection models using copulas with the `heckmancopula` command, see [Hasebe \(2013\)](#).

The methods in this chapter are highly parametric. Less parametric methods lead to partially identified models with bounds being placed using moment inequalities. For a recent survey, see [Molinari \(2020\)](#).

The `eintreg` and `eregress` commands, members of the class of ERM commands for recursive models, provide ML estimates of a wide range of models with endogenous regressors and sample selection; see section [23.7](#) for details.

19.13 Exercises

1. Consider the “linear version” of the tobit model used in this chapter. Using tests of homoskedasticity and normality, compare the outcome of the tests with those for the log version of the model.
2. Using the linear form of the tobit model in the preceding exercise, compare average predicted expenditure levels for those with insurance and those without insurance (`ins=0`). Compare these results with those from the tobit model for $\log(\text{ambexp})$.
3. Suppose we want to study the sensitivity of the predicted expenditure from the log form of the tobit model to neglected homoskedasticity. Observe from the table in section [19.8](#) that the prediction formula involves the variance parameter, σ^2 , that will be replaced by its estimate. Using the censoring threshold 0, draw a simulated heteroskedastic sample from a lognormal regression model with a single exogenous variable. Consider two levels of heteroskedasticity, low and high. By considering variations in the estimated σ^2 , show how the resulting biases in the estimate of σ^2 from the homoskedastic tobit model lead to biases in the mean prediction.
4. Repeat the simulation exercise using regression errors that are drawn from a $\chi^2(5)$ distribution. Recenter the simulated draws by subtracting the mean so that the recentered errors have a zero mean. Summarize the results of the prediction exercise for this case.
5. A conditional predictor for levels $E(y|\mathbf{x}, y > 0)$ mentioned in section 4.2, given parameters of a model fit in logs, is $\exp(\mathbf{x}'\hat{\boldsymbol{\beta}})N^{-1} \sum_i \exp(\hat{\varepsilon}_i)$. This expression is based on the assumption that ε_i are independent and identically distributed but normality is not assumed. Apply this conditional predictor to both the parameters of the two-part and selection models fit by the two-step procedure, and obtain estimates of $E(y|\mathbf{x}, y > 0)$ and $E(y|\mathbf{x})$. Compare the results with those given in section [19.8](#).
6. Repeat the calculations of scores, `gres1`, and `gres2` reported in section [19.4.6](#). Test that the calculations are done correctly; both score components should have a zero-mean property.
7. Make an unbalanced panel dataset by using the data of section 8.4 but then typing `set seed 10101` and `drop if runiform() < 0.2`. This will

randomly drop 20% of the individual–year observations. Type `xtdescribe`. Do you obtain the expected patterns of missing data? Use `xtsum` to describe the variation in `id`, `t`, `wage`, `ed`, and `south`. How do the results compare with those from the full panel? Use `xttab` and `xttrans` to provide interpretations of how `south` changes for individuals over time. Compare the within estimator with that in section 8.5 using the balanced panel.

8. Return to section [19.11.1](#), which analyzes the effect of IPW on the estimates. Only a subset of results is reproduced. Verify that the main conclusions of the exercise also apply to the estimators for which the results are not reported. The code is already provided.

Chapter 20

Count-data models

20.1 Introduction

In many contexts, the outcome of interest is a nonnegative integer, or a count, denoted by y , $y \in x\mathbb{N}_0 = \{0, 1, 2, \dots\}$. Examples can be found in demography, economics, ecology, environmental studies, insurance, and finance, to mention just a few of the areas of application.

The objective is to analyze y in a regression setting, given a vector of K covariates, \mathbf{x} . Because the response variable is discrete, its distribution places probability mass at nonnegative integer values only. Fully parametric formulations of count models accommodate this property of the distribution. Some semiparametric regression models only accommodate $y \geq 0$ but not discreteness. Count regressions are nonlinear; $E(y|\mathbf{x})$ is usually a nonlinear function, most commonly a single-index function like $\exp(\mathbf{x}'\boldsymbol{\beta})$. Several special features of count regression models are intimately connected to discreteness and nonlinearity.

If interest lies solely in modeling the conditional mean of y , and it is felt that a good model for this is $\exp(\mathbf{x}'\boldsymbol{\beta})$, then Poisson regression with inference based on appropriate robust standard errors is more than adequate. Furthermore, Poisson regression for modeling the conditional mean can be applied to dependent variables that are continuous and nonnegative, not just to counts. In particular, for right-skewed nonnegative data, Poisson regression can avoid the complications that can arise with ordinary least-squares (OLS) regression in logs when some observations take value zero. This use of Poisson regression is also called exponential regression.

Standard complications in analyzing count data include the following: the presence of unobserved heterogeneity akin to omitted variables; the small-mean property of y as manifested in the presence of many zeros, sometimes an “excess” of zeros; truncation in the observed distribution of y ; and endogenous regressors. To deal with these topics, one may have to use commands other than the `poisson` command. For models with fixed effects (FE) the Poisson FE model has the special property that, like the linear FE model, it does not suffer from the incidental parameters problem; see section [22.6.6](#).

The chapter begins with the basic Poisson and negative binomial (NB) models, using the `poisson` and `nbreg` commands, and then details some empirically important extensions, including the (double) hurdle or two-part, finite-mixture, and zero-inflated (or point-mass) models. The last part of the chapter deals with complications arising from endogenous regressors, clustered observations, and quantile regression.

20.2 Modeling strategies for count data

The natural starting point for analyses of counts is the Poisson distribution, with mean μ that is necessarily greater than zero because counts are nonnegative. For Poisson regression, the standard mean parameterization is $\mu = \exp(\mathbf{x}'\boldsymbol{\beta})$ to ensure that $\mu > 0$.

As explained below, the Poisson maximum-likelihood estimator (MLE) for $\boldsymbol{\beta}$ has the robustness property that its consistency requires only that the conditional mean be correctly specified; that is, that $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$. The data need not be Poisson distributed. This robustness property, enjoyed by few MLE, is analogous to consistency of the MLE in the linear model with independent homoskedastic normally distributed errors requiring only that $E(y|\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta}$.

This leads to two distinct modeling strategies for count data.

The first approach, adequate for many analyses, is a quasi-ML approach that performs Poisson regression using the `poisson` command. This is analogous to simply performing OLS regression for a continuous dependent variable. One complication, however, is that robust standard errors should always be used, as explained below.

The second approach is a fully parametric approach. This is necessary given complications such as predicting the probability that counts lie in a certain range or modeling count data that are truncated or censored. In that case, the Poisson distribution is inadequate, and one should use more general parametric models such as the NB or finite mixture models.

We first present the Poisson and NB distributions before providing more detail on the modeling strategies.

20.2.1 Generated Poisson data

The univariate Poisson distribution, denoted by $\text{Poisson}(y|\mu)$, for the number of occurrences of the event y over a fixed exposure period has the probability mass function

$$\Pr(Y = y) = \frac{e^{-\mu} \mu^y}{y!}, \quad y = 0, 1, 2, \dots \quad (20.1)$$

where μ is the intensity or rate parameter. The first two moments are

$$E(Y) = \mu$$

$$\text{Var}(Y) = \mu$$

This shows the well-known equality of mean and variance property, also called the equidispersion property, of the Poisson distribution. And it implies that in a Poisson regression $\text{Var}(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$, so the model is intrinsically heteroskedastic.

To illustrate some features of Poisson-distributed data, we use the `rpoisson()` function to make draws from the $\text{Poisson}(y|\mu = 1)$ distribution.

```
. * Poisson (mu=1) generated data
. qui set obs 10000
. set seed 10101           // Set the seed !
. generate xpois = rpoisson(1) // Draw from Poisson(mu=1)
. qui histogram xpois, discrete xtitle("Poisson") saving(histpois.gph, replace)
. summarize xpois
```

Variable	Obs	Mean	Std. dev.	Min	Max
xpois	10,000	.9989	1.004689	0	7

xpois	Freq.	Percent	Cum.
0	3,699	36.99	36.99
1	3,666	36.66	73.65
2	1,828	18.28	91.93
3	609	6.09	98.02
4	155	1.55	99.57
5	39	0.39	99.96
6	3	0.03	99.99
7	1	0.01	100.00
Total	10,000	100.00	

The expected frequency of zeros from (20.1) is $\Pr(Y = 0|\mu = 1) = e^{-1} = 0.368$. The simulated sample has nearly 37% zeros. Clearly, the larger is μ , the smaller will be the proportion of zeros; for example, $\Pr(Y = 0|\mu = 5) = 0.000067$.

For data with a small mean, as for example in the case of number of the children born in a family (or individual data on annual number of accidents or hospitalizations), zero observations are an important feature of the data. Further, when the mean is small, a high proportion of the sample will cluster on a relatively few distinct values. In this example, about 98% of the observations cluster on just four distinct values.

The generated data also reflect the equidispersion property, that is, equality of mean and variance of Y , because the standard deviation and hence variance are close to 1.

20.2.2 Overdispersion and NB data

The equidispersion property of the Poisson is commonly violated in applied work because overdispersion is common. Then the (conditional) variance exceeds the (conditional) mean. Such additional dispersion can be accounted for in many ways, of which the presence of unobserved heterogeneity is one of the most common.

Unobserved heterogeneity, which generates additional variability in y , can be generated by introducing multiplicative randomness. We replace μ with $\mu\nu$, where ν is a random variable, hence $y \sim \text{Poisson}(y|\mu\nu)$. Suppose we specify ν such that $E(\nu) = 1$ and $\text{Var}(\nu) = \sigma^2$. Then it is straightforward to show that ν preserves the mean but increases dispersion. Specifically, $E(y) = \mu$ and $\text{Var}(y) = \mu(1 + \mu\sigma^2) > E(y) = \mu$. The term “overdispersion” describes the feature $\text{Var}(y) > E(y)$ or, more precisely, $\text{Var}(y|\mathbf{x}) > E(y|\mathbf{x})$ in a regression model.

In the well-known special case that $\nu \sim \text{Gamma}(1, \alpha)$, where α is the variance parameter of the gamma distribution, the marginal distribution of y is a Poisson–gamma mixture with a closed form—the NB distribution denoted by $\text{NB}(\mu, \alpha)$ —whose probability mass function is

$$\Pr(Y = y|\mu, \alpha) = \frac{\Gamma(\alpha^{-1} + y)}{\Gamma(\alpha^{-1})\Gamma(y + 1)} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu} \right)^{\alpha^{-1}} \left(\frac{\mu}{\mu + \alpha^{-1}} \right)^y \quad (20.2)$$

where $\Gamma(\cdot)$ denotes the gamma integral that specializes to a factorial for an integer argument. The NB model is more general than the Poisson model because it accommodates overdispersion and it reduces to the Poisson model as $\alpha \rightarrow 0$. The moments of the NB2 are $E(y|\mu, \alpha) = \mu$ and $\text{Var}(y|\mu, \alpha) = \mu(1 + \alpha\mu)$. Empirically, the quadratic variance function is a versatile approximation in a wide variety of cases of overdispersed data.

The NB regression model lets $\mu = \exp(\mathbf{x}'\boldsymbol{\beta})$ and leaves α as a constant. The default option for the NB regression in Stata is the version with a quadratic variance (NB2). Another variant of NB in the literature has a linear variance function, $\text{Var}(y|\mu, \alpha) = (1 + \alpha)\mu$, and is called the NB1 model. See [Cameron and Trivedi \(2005, chap. 20.4\)](#).

Using the mixture interpretation of the NB model, we simulate a sample from the $\text{NB}(\mu = 1, \alpha = 1)$ distribution. We first use the `rgamma(1, 1)` function to obtain the gamma draw, v , with a mean of $1 \times 1 = 1$ and a variance of $\alpha = 1 \times 1^2 = 1$; see section 5.2.4. We then obtain Poisson draws with $\mu v = 1 \times v = v$, using the `rpoisson()` function with the argument v .

```
. * Negative binomial (mu=1 var=2) generated data
. set seed 10101 // Set the seed !
. generate xg = rgamma(1,1)
. generate xnegin = rpoisson(xg) // NB generated as a Poisson-gamma mixture
. qui histogram xnegin, discrete xtitle("Negative binomial")
>     saving(histnb2.gph, replace)
. summarize xnegin
```

Variable	Obs	Mean	Std. dev.	Min	Max
xnegin	10,000	1.0044	1.422949	0	15

```
. tabulate xnegin
```

xnegin	Freq.	Percent	Cum.
0	4,975	49.75	49.75
1	2,501	25.01	74.76
2	1,282	12.82	87.58
3	625	6.25	93.83
4	319	3.19	97.02
5	143	1.43	98.45
6	75	0.75	99.20
7	33	0.33	99.53
8	22	0.22	99.75
9	9	0.09	99.84
10	9	0.09	99.93
11	4	0.04	99.97
13	2	0.02	99.99
15	1	0.01	100.00
Total	10,000	100.00	

As expected, the mean is close to 1 and the variance of $1.42^2 = 2.01$ is close to $(1 + 1) \times 1 = 2$. Relative to the Poisson(1) pseudorandom draws, this sample has more 0s, a longer right tail, and a variance-to-mean ratio in excess of 1. These features are a consequence of introducing the multiplicative heterogeneity term.

The `rnbomial()` function can instead be used to make direct draws from the NB distribution, but because it uses an alternative parameterization of the NB distribution, it is easier to use the above Poisson–gamma mixture.

20.2.3 Conditional mean modeling strategies

Given (20.1), $\mu = \exp(\mathbf{x}'\boldsymbol{\beta})$, and the assumption that the observations $(y_i | \mathbf{x}_i)$ are independent, Poisson ML is often the starting point of a modeling exercise, especially if the entire distribution and not just the conditional mean is the object of interest.

Count data are often overdispersed. One approach is to maintain the conditional mean assumption $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$. Then one can continue to use the Poisson MLE, which retains its consistency. However, one must then relax the equivariance assumption and obtain a robust estimate of the

variance–covariance matrix of the estimator. For independent data, one obtains heteroskedastic-robust standard errors using the `vce(robust)` option. For within-cluster correlation, one instead obtains cluster-robust standard errors using the `vce(cluster clustvar)` option.

Generalized linear models are a class of models for which the conditional mean modeling approach can be applied. The Poisson model is a member of this class, so Poisson regression can also be performed as a special case of the `glm` command. And the conditional mean approach is the basis for generalized method of moments (GMM) estimation of models with endogenous regressors.

The NB2 model, the default for the `nbreg` command, also has the property that consistency requires only $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$. Thus, either the Poisson model or the NB2 model could be used if $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$. While the NB2 model estimator may be more efficient, in practice, the efficiency gains can be slight, and it is common to simply use the Poisson model if interest lies in modeling only the conditional mean and not conditional probabilities.

20.2.4 Poisson regression for continuous nonnegative dependent variable

Poisson regression is not restricted to count data and is applicable to continuous nonnegative data where a good model for the conditional mean is $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$. Thus, Poisson regression is often called exponential regression.

In particular, the lognormal model is often used for continuous right-skewed positive data, with $E(\ln y|\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta}$. This has the limitation of not being applicable if some values of $y = 0$. One approach is to use the two-part model of section 19.5. A simpler approach is to perform Poisson regression in a model with $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$, basing inference on robust standard errors. This requires the assumption that zero observations come from the same process as positive observations. This model also permits direct prediction of $E(y|\mathbf{x})$ without the complication of retransformation bias.

A leading application is to model trade volume between countries using the gravity model of trade; see [Santos Silva and Tenreyro \(2006\)](#).

20.2.5 Fully parametric modeling strategies

A fully parametric approach begins with the NB2 model. For example, this is a starting point if predicted probabilities are desired.

But complications such as censoring or truncation or different treatment of zero counts lead to $E(y|x) \neq \exp(x'\beta)$. The empirical examples in sections [20.4](#) to [20.6](#) illustrate several extensions of simple Poisson or NB2 regression.

20.3 Poisson and negative binomial models

In this section, we fit Poisson and NB2 count-data models for the annual number of doctor visits (`docvis`).

20.3.1 Data summary

The data are a cross-sectional sample from the U.S. Medical Expenditure Panel Survey for 2003. We model the annual number of doctor visits (`docvis`) using a sample of the Medicare population aged 65 and higher.

The covariates in the regressions are age (`age`), squared age (`age2`), years of education (`educyr`), presence of activity limitation (`actlim`), number of chronic conditions (`totchr`), having private insurance that supplements Medicare (`private`), and having public Medicaid insurance for low-income individuals that supplements Medicare insurance (`medicaid`).

Summary statistics for the dependent variable and regressors are as follows:

```
. * Summary statistics for doctor-visits data
. qui use mus220mepsdocvis, clear
. global xlist private medicaid age age2 educyr actlim totchr
. summarize docvis $xlist
```

Variable	Obs	Mean	Std. dev.	Min	Max
docvis	3,677	6.822682	7.394937	0	144
private	3,677	.4966005	.5000564	0	1
medicaid	3,677	.166712	.3727692	0	1
age	3,677	74.24476	6.376638	65	90
age2	3,677	5552.936	958.9996	4225	8100
educyr	3,677	11.18031	3.827676	0	17
actlim	3,677	.333152	.4714045	0	1
totchr	3,677	1.843351	1.350026	0	8

The sampled individuals are aged 65–90 years, and a considerable portion has an activity limitation or chronic condition. The sample mean of `docvis` is 6.82, and the sample variance is $7.39^2 = 54.61$, so there is great overdispersion.

For count data, one should always obtain a frequency distribution or histogram. To reduce output, we create a variable, `dvrang`, with counts of 11–40

recoded as 40 and counts of 41–143 recoded as 143. We have

```
. * Tabulate docvis after recoding values > 10 to ranges 11-40 or 41-143
. generate dvrage = docvis
. recode dvrage (11/40 = 40) (41/143 = 143)
(786 changes made to dvrage)
. tabulate dvrage
```

dvrage	Freq.	Percent	Cum.
0	401	10.91	10.91
1	314	8.54	19.45
2	358	9.74	29.18
3	334	9.08	38.26
4	339	9.22	47.48
5	266	7.23	54.72
6	231	6.28	61.00
7	202	5.49	66.49
8	179	4.87	71.36
9	154	4.19	75.55
10	108	2.94	78.49
40	774	21.05	99.54
143	16	0.44	99.97
144	1	0.03	100.00
Total	3,677	100.00	

The distribution has a long right tail, 22% of observations exceed 10, and the maximum is 144. More than 99% of the values are under 40. The proportion of zeros is 10.9%. This is relatively low for these types of data, partly because the data pertain to the elderly population. Samples of the younger and usually healthier population often have as many as 90% zero observations for some health outcomes.

20.3.2 Poisson model

For the Poisson model, the probability mass function is the Poisson distribution given in (20.1), and the default is the exponential mean parameterization

$$\mu_i = \exp(\mathbf{x}'_i \boldsymbol{\beta}), \quad i = 1, \dots, N \tag{20.3}$$

where by assumption there are K linearly independent covariates, usually including a constant. This specification restricts the conditional mean to be positive.

The Poisson MLE, denoted by $\hat{\beta}_P$, is the solution to K nonlinear equations corresponding to the ML first-order conditions

$$\sum_{i=1}^N \{y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})\} \mathbf{x}_i = \mathbf{0} \quad (20.4)$$

If \mathbf{x}_i includes a constant term, then the residuals $y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})$ sum to zero based on (20.4). Because the log-likelihood function is globally concave, the iterative solution algorithm, usually the Newton–Raphson (see section 16.2), converges fast to a unique global maximum.

An informal approach to consistency is that it requires that the sample moment conditions defining an estimator should hold in expectation in the population. Thus, consistency requires that the left-hand side of (20.4) have expected value $\mathbf{0}$. A sufficient condition is that $E(y_i | \mathbf{x}_i) = \exp(\mathbf{x}'_i \boldsymbol{\beta})$. So consistency of the Poisson MLE requires only that the functional form for the conditional mean be correctly specified.

By standard ML theory, if the Poisson model is parametrically correctly specified, the estimator $\hat{\beta}_P$ is consistent for $\boldsymbol{\beta}$, with a covariance matrix estimated by

$$\hat{V}(\hat{\beta}_P) = \left(\sum_{i=1}^N \hat{\mu}_i \mathbf{x}_i \mathbf{x}'_i \right)^{-1} \quad (20.5)$$

where $\hat{\mu}_i = \exp(\mathbf{x}'_i \hat{\beta}_P)$. We show below that it is usually very misleading to use (20.5) to estimate the variance–covariance matrix of the estimator (vce) of $\hat{\beta}_P$, and it is better to use robust standard errors given in (20.6).

The Poisson MLE is implemented with the `poisson` command, and the default estimate of the vce is that in (20.5). The syntax for `poisson`, similar to that for

regress, is

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

The `vce(robust)` option yields a robust estimate of the VCE.

Two commonly used options are `offset()` and `exposure()`. Suppose regressor z is an exposure variable, such as time. Then, as z doubles, we expect the count, y , to double. Then $E(y|z, \mathbf{x}_2) = z \exp(\mathbf{x}'_2 \boldsymbol{\beta}) = \exp(\ln z + \mathbf{x}'_2 \boldsymbol{\beta})$. If the variable z appears in the regressor list, this constraint is imposed by using the `offset(z)` option. If instead the variable $\ln z = \ln(z)$ appears in the regressor list, this constraint is imposed by using the `exposure(lnz)` option.

Poisson model results

We first obtain and discuss the results for Poisson ML estimation.

```
. * Poisson with default ML standard errors
. poisson docvis $xlist, nolog
```

Poisson regression

Number of obs = 3,677
LR chi2(7) = 4477.98
Prob > chi2 = 0.0000
Pseudo R2 = 0.1297

Log likelihood = -15019.64

docvis	Coefficient	Std. err.	z	P> z	[95% conf. interval]
private	.1422324	.0143311	9.92	0.000	.114144 .1703208
medicaid	.0970005	.0189307	5.12	0.000	.0598969 .134104
age	.2936722	.0259563	11.31	0.000	.2427988 .3445457
age2	-.0019311	.0001724	-11.20	0.000	-.0022691 -.0015931
educyr	.0295562	.001882	15.70	0.000	.0258676 .0332449
actlim	.1864213	.014566	12.80	0.000	.1578726 .2149701
totchr	.2483898	.0046447	53.48	0.000	.2392864 .2574933
_cons	-10.18221	.9720115	-10.48	0.000	-12.08732 -8.277101

The sign of the coefficients gives the sign of marginal effects (MES). So on average, `docvis` is increasing in education, number of chronic conditions, being limited in activity, and having either type of supplementary health insurance. These results are consistent with a priori expectations. The effect of age is more complicated because it appears quadratically.

Exactly the same coefficient estimates are obtained using the `glm` command; see section 13.3.8.

The top part of the `poisson` output lists sample size, the likelihood-ratio (LR) test for the joint significance of the seven regressors, the *p*-value associated with the test, and the pseudo- R^2 statistic, which is intended to serve as a measure of the goodness of fit of the model (see section 13.8.1).

An alternative measure of the fit of the model is the squared coefficient of correlation between the fitted and observed values of the dependent variable. This is not provided by `poisson` but is easily computed as follows:

```
. * Poisson: Squared correlation between y and yhat
. predict yhat, n
. qui correlate docvis yhat
. display "Squared correlation between y and yhat = " r(rho)^2
Squared correlation between y and yhat = .1530784
```

The squared correlation coefficient is low but reasonable for cross-sectional data.

The variables in the Poisson model appear to be highly statistically significant, but this is partly due to great underestimation of the standard errors, as we explain next.

Heteroskedastic-robust estimate of VCE for Poisson MLE

As explained after (20.4) the Poisson MLE retains consistency provided that the conditional mean function in (20.3) is correctly specified. The dependent variable `y` need not be actually Poisson distributed and need not be a count variable.

When the dependent variable is not Poisson distributed, but the conditional mean function is specified by (20.3), we can use the pseudo-ML or quasi-ML approach (see section 13.3.1), which maximizes the Poisson likelihood function but uses the robust estimate of the VCE,

$$\widehat{V}_{\text{Rob}} \left(\widehat{\beta}_P \right) = \left(\sum_{i=1}^N \widehat{\mu}_i \mathbf{x}_i \mathbf{x}'_i \right)^{-1} \left\{ \sum_{i=1}^N (y_i - \widehat{\mu}_i)^2 \mathbf{x}_i \mathbf{x}'_i \right\} \left(\sum_{i=1}^N \widehat{\mu}_i \mathbf{x}_i \mathbf{x}'_i \right)^{-1} \quad (20.6)$$

where $\widehat{\mu}_i = \exp(\mathbf{x}'_i \widehat{\beta}_P)$. That is, we use the Poisson MLE to obtain our point estimates, but we obtain robust estimates of the VCE. With overdispersion, the

variances will be larger using (20.6) than (20.5) because with equidispersion (20.6) reduces to (20.5), but with overdispersion, $(y_i - \hat{\mu}_i)^2 > \hat{\mu}_i$, on average. In the rare case of underdispersion, this ordering is reversed.

This preferred estimate of the VCE is obtained by using the `vce(robust)` option of `poisson`. We obtain

```
. * Poisson with robust standard errors
. poisson docvis $xlist, vce(robust) nolog // Poisson robust SEs
Poisson regression
Number of obs = 3,677
Wald chi2(7) = 720.43
Prob > chi2 = 0.0000
Pseudo R2 = 0.1297
Log pseudolikelihood = -15019.64
```

docvis	Coefficient	Robust		z	P> z	[95% conf. interval]
		std. err.				
private	.1422324	.036356	3.91	0.000	.070976	.2134889
medicaid	.0970005	.0568264	1.71	0.088	-.0143773	.2083783
age	.2936722	.0629776	4.66	0.000	.1702383	.4171061
age2	-.0019311	.0004166	-4.64	0.000	-.0027475	-.0011147
educyr	.0295562	.0048454	6.10	0.000	.0200594	.039053
actlim	.1864213	.0396569	4.70	0.000	.1086953	.2641474
totchr	.2483898	.0125786	19.75	0.000	.2237361	.2730435
_cons	-10.18221	2.369212	-4.30	0.000	-14.82578	-5.538638

Compared with the Poisson MLE standard errors, the robust standard errors are two to three times larger. This is a very common feature of results for Poisson regression applied to overdispersed data, and microeconometrics count data are often considerably overdispersed.

Cluster-robust estimate of VCE for Poisson MLE

For observations correlated within cluster, and independent across cluster, we instead obtain cluster-robust standard errors using the `vce(cluster clustvar)` option. This corrects for both overdispersion and clustering.

Test of overdispersion

For completeness, we present a formal test of equidispersion, even though common practice for Poisson regression is to always obtain robust standard errors that are valid even if data are overdispersed or underdispersed. (Similarly, it is common to always use robust standard errors after OLS regression, rather than first testing whether errors are homoskedastic.)

A formal test of the null hypothesis of equidispersion, $\text{Var}(y|\mathbf{x}) = E(y|\mathbf{x})$, against the alternative of overdispersion can be based on the equation

$$\text{Var}(y|\mathbf{x}) = E(y|\mathbf{x}) + \alpha^2 E(y|\mathbf{x})$$

which is the variance function for the NB2 model. We test $H_0: \alpha = 0$ against $H_1: \alpha > 0$.

The test can be implemented by an auxiliary regression of the generated dependent variable, $\{(y - \hat{\mu})^2 - y\}/\hat{\mu}$ on $\hat{\mu}$, without an intercept term, and performing a t test of whether the coefficient of $\hat{\mu}$ is zero; see [Cameron and Trivedi \(2005\)](#), 670–671) for details of this and other specifications of overdispersion.

```
. * Overdispersion test against V(y|x) = E(y|x) + a*{E(y|x)^2}
. qui poisson docvis $xlist, vce(robust)
. predict muhat, n
. qui generate ystar = ((docvis-muhat)^2 - docvis)/muhat
. regress ystar muhat, noconstant noheader
```

ystar	Coefficient	Std. err.	t	P> t	[95% conf. interval]
muhat	.7047319	.1035926	6.80	0.000	.5016273 .9078365

The outcome indicates the presence of significant overdispersion. One way to model this feature of the data is to use the NB model. More common is to simply use `poisson` with the `vce(robust)` option.

Pearson and deviance goodness-of-fit tests

The Stata postestimation command `estat gof` provides two goodness-of-fit tests after the `poisson` command, though not after other count regression commands.

Pearson's test statistic is for testing variance-mean equality and equals $\sum_{i=1}^N (y_i - \hat{\mu}_i)^2 / \hat{\mu}_i$, where $\hat{\mu}_i = \exp(\mathbf{x}'_i \hat{\beta}_P)$. The deviance statistic is a measure used in the generalized linear models literature that is defined in section 13.8.3.

We apply the command to the Poisson regression model reported above.

```

. * Pearson and deviance measures after Poisson
. qui poisson docvis $xlist, nolog
. estat gof
    Deviance goodness-of-fit = 18395.14
    Prob > chi2(3669)      = 0.0000
    Pearson goodness-of-fit = 23147.38
    Prob > chi2(3669)      = 0.0000

```

Both statistics strongly reject the Poisson model as a model of the conditional distribution.

Nonetheless, the Poisson model is fine for modeling the conditional mean, provided the conditional mean is correctly specified. But inference needs to be based on robust standard errors.

Coefficient interpretation and MEs

Section 13.7 discusses coefficient interpretation and MEs estimation, both in general and for the exponential conditional mean, $\exp(\mathbf{x}'\boldsymbol{\beta})$.

The coefficients can be interpreted as a semielasticity because the conditional mean function is of exponential form; see section 13.7.3. Thus, the coefficient of `educyr` of 0.030 can be interpreted as one more year of education being associated with a 3.0% increase in the number of doctor visits. The `irr` option of `poisson` produces exponentiated coefficients, $e^{\hat{\beta}}$, that can be given a multiplicative interpretation. Thus, one more year of education is associated with doctor visits increasing by the multiple $e^{0.030} \approx 1.030$.

The ME of a unit change in a continuous regressor, x_j , equals $\partial E(y|\mathbf{x})/\partial x_j = \beta_j \exp(\mathbf{x}'\boldsymbol{\beta})$, which depends on the evaluation point, \mathbf{x} . From section 13.7.2, there are three standard ME measures.

It can be shown that the average marginal effect (AME) for the Poisson model with an intercept equals $\hat{\beta}_j \bar{y}$. For example, one more year of education is associated with $0.02956 \times 6.823 = 0.2017$ additional doctor visits. The same result, along with a confidence interval, can be obtained by using the `margins, dydx()` command.

To ensure that MEs for binary regressors are calculated using the finite-difference method, we use the factor-variable `i.` operator in defining the model

to be fit by the `poisson` command. And to obtain the ME with respect to age, which enters quadratically, we use the factor-variable `##` operator. We obtain

```
. * Poisson AMEs
. qui poisson docvis i.private i.medicaid c.age##c.age educyr i.actlim
>      totchr, vce(robust)
. margins, dydx(*)

Average marginal effects                                         Number of obs = 3,677
Model VCE: Robust

Expression: Predicted number of events, predict()
dy/dx wrt: 1.private 1.medicaid age educyr 1.actlim totchr
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.private	.9701906	.2473149	3.92	0.000	.4854622	1.454919
1.medicaid	.6830664	.4153252	1.64	0.100	-.130956	1.497089
age	.0385842	.0172075	2.24	0.025	.0048581	.0723103
educyr	.2016526	.0337805	5.97	0.000	.1354441	.2678612
1.actlim	1.295942	.2850588	4.55	0.000	.7372367	1.854647
totchr	1.694685	.0908883	18.65	0.000	1.516547	1.872823

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

For example, one more year of education is associated with 0.202 additional doctor visits. The output also provides confidence intervals for the ME. The ME at the mean (MEM) is calculated with the `atmeans` option of `margins`, and the ME at a representative value (MER) is calculated with the `at()` option.

20.3.3 NB2 model

The NB2 model with a quadratic variance function is consistent with overdispersion generated by a Poisson–gamma mixture (see section [20.2.2](#)), but it can also be considered simply as a more flexible functional form for overdispersed count data.

The NB2 model MLE, denoted by $\hat{\beta}_{\text{NB2}}$, maximizes the log likelihood based on the probability mass function ([20.2](#)), where again $\mu = \exp(\mathbf{x}'\boldsymbol{\beta})$, whereas α is simply a constant parameter. The estimators $\hat{\beta}_{\text{NB2}}$ and $\hat{\alpha}_{\text{NB2}}$ are the solution to the $K + 1$ nonlinear equations corresponding to the ML first-order conditions

(20.7)

$$\sum_{i=1}^N \frac{y_i - \mu_i}{1 + \alpha\mu_i} \mathbf{x}_i = \mathbf{0}$$

$$\sum_{i=1}^N \left[\frac{1}{\alpha^2} \left\{ \ln(1 + \alpha\mu_i) - \sum_{j=0}^{y_i-1} \frac{1}{(j + \alpha^{-1})} \right\} + \frac{y_i - \mu_i}{\alpha(1 + \alpha\mu_i)} \right] = 0$$

The K -element β equations, the first line in (20.7), are in general different from (20.4) and are sometimes harder to solve using the iterative algorithms. Very large or small values of α can generate numerical instability, and convergence of the algorithm is not guaranteed.

Analogous to the discussion for the Poisson MLE, consistency of the NB2 MLE requires that the left-hand side of the first equation in (20.7) have expected value 0. A sufficient condition is again that $E(y_i|\mathbf{x}_i) = \exp(\mathbf{x}'_i\beta)$. So consistency of the NB2 MLE requires only that the functional form for the conditional mean be correctly specified.

While the NB2 and Poisson models have the same functional form for the conditional mean, the fitted probability distribution of the NB2 can be quite different from that of the Poisson because of different models for the conditional variance. If the data are indeed overdispersed, then the NB2 model is preferred if the goal is to model the probability distribution and not just the conditional mean.

The NB2 model is not a panacea. There are other reasons for overdispersion, including misspecification due to restriction to an exponential conditional mean. Alternative models are presented in sections 20.4–20.6.

The partial syntax for the MLE for the NB model is similar to that for the `poisson` command:

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

The default fits an NB2 model, and the `dispersion(constant)` option fits an NB1 model.

As noted in section 13.3.1, we use robust standard errors, unless there is reason not to do so. In practice, with independent data, the difference between default standard errors and those obtained using the `vce(robust)` option is not great, because overdispersion is reasonably modeled, and is much less than the

difference following Poisson regression. If data are clustered, then the `vce(cluster clustvar)` option should be used.

NB2 model results

Given the presence of considerable overdispersion in our data, the NB2 model should be considered. We obtain

```
. * NB2: Standard negative binomial with robust standard errors
. nbreg docvis $xlist, vce(robust) nolog

Negative binomial regression                                         Number of obs = 3,677
Dispersion: mean                                                 Wald chi2(7)  = 725.69
Log pseudolikelihood = -10589.339                               Prob > chi2   = 0.0000
                                                               Pseudo R2    = 0.0352
```

docvis	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
private	.1640928	.0368851	4.45	0.000	.0917993 .2363864
medicaid	.100337	.0567414	1.77	0.077	-.0108742 .2115482
age	.2941294	.0646326	4.55	0.000	.1674518 .4208069
age2	-.0019282	.0004283	-4.50	0.000	-.0027677 -.0010886
educyr	.0286947	.0049211	5.83	0.000	.0190494 .03834
actlim	.1895376	.0393922	4.81	0.000	.1123304 .2667449
totchr	.2776441	.013244	20.96	0.000	.2516864 .3036019
_cons	-10.29749	2.424128	-4.25	0.000	-15.04869 -5.546284
/lnalpha	-.4452773	.0377998			-.5193636 -.371191
alpha	.6406466	.0242163			.594899 .6899121

The parameter estimates are all within 15% of those for the Poisson MLE and are often much closer than this. The robust standard errors are very similar to those from Poisson regression; in practice, there is often little efficiency improvement in moving from Poisson to NB estimation of overdispersed count data. The parameters and MES are interpreted in the same way as for a Poisson model because both models have the same conditional mean.

The NB2 estimate of the overdispersion parameter of 0.64 is similar to the 0.70 from the auxiliary regression used in testing for overdispersion. Because $\alpha > 0$ necessarily, a test of overdispersion is a one-sided test that strongly rejects H_0 because the Wald statistic equals $0.6406/0.0242 = 26.5 > 1.645$ at level 0.05. If default standard errors are used, then the computer output also includes an LR test of $H_0: \alpha = 0$; an NB example is given in section 11.4.1.

The pseudo- R^2 is 0.035 compared with the 0.130 for the Poisson model. This difference, a seemingly worse fit than for the Poisson model, is because the pseudo- R^2 is not directly comparable across classes of models, here NB2 and Poisson.

More directly comparable is the squared correlation between fitted and actual counts. We obtain

```
. * NB2: Squared correlation between y and yhat
. predict ynbhat, n
. qui correlate docvis ynbhat
. display "Squared correlation between y and yhat = " r(rho)^2
Squared correlation between y and yhat = .14979846
```

This is similar to the 0.153 for the Poisson model, so the two models provide a similar fit for the conditional mean. The real advantage of the NB2 model is in fitting probabilities because this relies on specification of the density and not just the conditional mean.

20.3.4 Fitted probabilities for Poisson and NB2 models

To get more insight into the improvement in the fit, we should compare what the parameter estimates from the Poisson and the NB2 models imply for the fitted probability distribution of `docvis`.

Fitted probabilities for each observation and average fitted probabilities

Given an estimated parametric model, such as the Poisson or NB2, fitted probabilities for each individual and for a particular count value k , denoted $\widehat{p}_i(k)$, can be generated using the `pr()` option of the `predict` postestimation command. The same option can give predicted probabilities for a range of count values.

Average predicted probabilities

$$\bar{\widehat{p}}(k) = \frac{1}{N} \sum_{i=1}^N \widehat{p}_i(k), \quad k = 0, 1, 2, \dots \quad (20.8)$$

can be obtained using the `margins` command with the `predict(pr())` option.

For example, command `predict phat2, pr(2)` generates variable $\hat{p}_i(2)$ for each observation, and command `margins, predict(pr(2))` reports $\bar{\hat{p}}(2)$ along with a standard error and 95% confidence interval.

We additionally present some convenient community-contributed commands.

The `countfit` command to compare fitted and actual cell frequencies

The community-contributed `countfit` command ([Long and Freese 2014](#)) compares the average predicted probabilities $\bar{\hat{p}}(k)$ defined in (20.8) with the actual cell frequencies, for a range of values of k . The `prm` option fits the Poisson model, and the `nbreg` option fits the NB2 model. Additional options control the amount of output produced by the command. In particular, the `maxcount(#)` option sets the maximum count for which predicted probabilities are evaluated; the default is `maxcount(9)`.

For the Poisson model, we obtain

```
. * Poisson: Sample versus avg predicted probabilities of y = 0, 1, ..., 5
. countfit docvis $xlist, maxcount(5) prm nograph noestimatesnofit
Comparison of Mean Observed and Predicted Count

```

Model	Maximum Difference	At Value	Mean Diff
PRM	0.102	0	0.045

```
PRM: Predicted and actual probabilities

```

Count	Actual	Predicted	Diff	Pearson
0	0.109	0.007	0.102	5168.233
1	0.085	0.030	0.056	387.868
2	0.097	0.063	0.034	69.000
3	0.091	0.095	0.005	0.789
4	0.092	0.116	0.024	17.861
5	0.072	0.121	0.049	72.441
Sum	0.547	0.432	0.269	5716.192

The Poisson model seriously underestimates the probability mass at low counts. In particular, the predicted probabilities at 0 and 1 counts are 0.007 and 0.030 compared with sample frequencies of 0.109 and 0.085.

For the NB2 model, which allows for overdispersion, we obtain

```

. * NB2: Sample versus average predicted probabilities of y = 0, 1, ..., 5
. countfit docvis $xlist, maxcount(5) nbreg nograph noestimates nofit
Comparison of Mean Observed and Predicted Count

          Maximum      At      Mean
Model     Difference    Value    |Diff|
-----
NBRM      -0.023        1       0.010

NBRM: Predicted and actual probabilities

Count   Actual     Predicted    |Diff|  Pearson
-----
0       0.109      0.091      0.018   12.708
1       0.085      0.108      0.023   17.288
2       0.097      0.105      0.008   2.270
3       0.091      0.096      0.005   1.086
4       0.092      0.085      0.007   2.333
5       0.072      0.074      0.001   0.072
-----
Sum     0.547      0.559      0.062   35.757

```

The fit is now much better. The greatest discrepancy is for $y = 1$, with a predicted probability of 0.108, which exceeds the sample frequency of 0.085.

The comparison confirms that the NB2 model provides a much better fit of the probabilities than the Poisson model (even though for the conditional mean, the MES are similar for the two models).

The final column, marked `Pearson`, gives N times $(\text{Diff})^2 / \text{Predicted}$, where `Diff` is the difference between average fitted and empirical frequencies, for each value of `docvis` up to that given by the `maxcount()` option. Although these values are a good rough indicator of goodness of fit, caution should be exercised in using these numbers as the basis of a Pearson chi-squared goodness-of-fit test because the fitted probabilities are functions of estimated coefficients; see [Cameron and Trivedi \(2005, 266\)](#).

The `chi2gof` command to test difference between fitted and actual cell frequencies

The community-contributed `chi2gof` command of [Manjón and Martínez \(2014\)](#) performs the correct chi-squared goodness-of-fit test that allows for estimation error in obtaining the fitted probabilities.

Applying this to the preceding NB2 example, we obtain

```

. * NB2: Formal chisquare goodness of fit test
. qui nbreg docvis $xlist
. chi2gof, cells(0, 1, 2, 3, 4, 5) table
Chi-square Goodness-of-Fit Test for NegBin Model:

    Chi-square chi2(6) = 41.82
    Prob>chi2 = 0.00

```

Cells	Abs. Freq.	Fitted			Abs. Dif.
		Rel. Freq.	Rel. Freq.	Abs. Freq.	
[0 to 0]	401	.1091	.0913	.0178	
[1 to 1]	314	.0854	.1079	.0225	
[2 to 2]	358	.0974	.1054	.0081	
[3 to 3]	334	.0908	.0962	.0053	
[4 to 4]	339	.0922	.0849	.0073	
[5 to 5]	266	.0723	.0735	.0012	
6 or more	1434	.4528	.4408	.012	

The same actual and fitted relative frequencies are given, as expected. Controlling for estimation error has led to a modest increase in the chi-squared test statistic from 35.76 to 41.82. The NB2 model is rejected, though as already noted it is a substantial improvement on the Poisson model.

The prcounts command to compute average fitted probabilities and cumulative probabilities

The community-contributed `prcounts` command ([Long and Freese 2014](#)) computes cumulative average fitted probabilities in addition to the average fitted predicted probabilities that are given by the community-contributed `countfit` command.

We use the `max(3)` option to reduce the amount of output.

```

. * NB2: Cumulative fitted probabilities and ave fitted probs of y = 0 to max()
. qui nbreg docvis $xlist, vce(robust)
. prcounts erpr, max(3)
. summarize erp*

```

Variable	Obs	Mean	Std. dev.	Min	Max
erprrate	3,677	6.890034	3.486562	2.078925	41.31503
erprpr0	3,677	.0912936	.0446796	.0056767	.2667141
erprpr1	3,677	.1079217	.0436249	.0085383	.2377842
erprpr2	3,677	.1054288	.0343211	.0105349	.1739022
erprpr3	3,677	.0961651	.024424	.0120494	.1266245
erprcu0	3,677	.0912936	.0446796	.0056767	.2667141
erprcu1	3,677	.1992154	.0881782	.0142149	.5044982
erprcu2	3,677	.3046441	.1220802	.0247498	.6784004
erprcu3	3,677	.4008093	.1455686	.0367992	.7962972
erprprgt	3,677	.5991907	.1455686	.2037028	.9632007

The output begins with the `erprrate` variable, which is the fitted mean and has an average value of 6.890, close to the sample mean of 6.823. The `erprpr0`–`erprpr3` variables are predictions of $\Pr(y_i = j)$, $j = 0, 1, 2, 3$, that have averages of 0.091, 0.108, 0.105, and 0.096, the same as given by the `countfit` command. The `erprcu0`–`erprcu3` variables are the corresponding cumulative predicted probabilities of 0.092, 0.199, 0.305, and 0.401.

The `prvalue` command to predict probabilities at a given value of the regressors

The community-contributed `prvalue` command ([Long and Freese 2014](#)) predicts probabilities for given values of the regressors, computed using $\hat{p}(k|\mathbf{x} = \mathbf{x}^*)$.

As an example, we obtain predicted probabilities for a person with private insurance and access to Medicaid, with other regressors set to their sample means. The `prvalue` command, with options used to minimize the length of output, following the `nbreg` command, yields

```

. * NB2: Predicted NB2 probabilities at x = x* of y = 0, 1, ..., 5
. qui nbreg docvis $xlist, vce(robust)
. prvalue, x(private=1 medicaid=1) max(5) brief
nbreg: Predictions for docvis
                                         95% Conf. Interval
Rate:                      7.34  [ 6.5004,     8.1796]
Pr(y=0|x):                 0.0660 [ 0.0563,     0.0758]
Pr(y=1|x):                 0.0850 [ 0.0742,     0.0958]
Pr(y=2|x):                 0.0898 [ 0.0802,     0.0994]
Pr(y=3|x):                 0.0879 [ 0.0802,     0.0955]
Pr(y=4|x):                 0.0826 [ 0.0771,     0.0882]
Pr(y=5|x):                 0.0758 [ 0.0722,     0.0793]

```

These predicted probabilities at a specific value of the regressors are within 30% of the average predicted probabilities for the NB2 model previously computed by using the `countfit` command.

Predicted probabilities and their MEs using the margins command

Predicted probabilities, with standard errors and confidence intervals, can be obtained using the `margins` command. For example,

```

* NB2: Predicted probabilities and their MEs using margins
. margins, predict(pr(0)) predict(pr(1)) at(private=1 medicaid=1) atmeans

```

gives the preceding predicted probabilities at $y = 0$ and $y = 1$. Adding the `dydx()` option will give the marginal effects.

Discussion

The assumption of gamma heterogeneity underlying the mixture interpretation of the NB2 model is very convenient, but there are other alternatives. For example, one can assume that heterogeneity is lognormally distributed. This specification does not lead to an analytical expression for the mixture distribution and requires an estimation method involving one-dimensional numerical integration.

Estimation using the `gsem` command is given in section [23.6.4](#).

20.3.5 Generalized NB model

The generalized NB model is an extension of the NB2 model that permits additional parameterization of the overdispersion parameter, α , in (20.2), where it is simply a positive constant in the NB2 model. The overdispersion parameter

can then vary across individuals, and the same variable can affect both the location and the scale parameters of the distribution, complicating the computation of MES. Alternatively, the model may be specified such that different variables may separately affect the location and scale of the distribution.

Even though, in principle, flexibility is desirable, such models are currently not widely used. The parameters of the model can be fit using the `gnbreg` command, which has a syntax similar to that of `nbreg`, with the addition of the `lnalpha()` option to specify the variables in the model for $\ln(\alpha)$.

We parameterize $\ln(\alpha)$ for the dummy variables `female` and `bh` (black/Hispanic).

```
. * Generalized negative binomial with alpha parameterized
. gnbreg docvis $xlist, lnalpha(female bh) vce(robust) nolog
Generalized negative binomial regression                               Number of obs = 3,677
                                                               Wald chi2(7) = 703.04
                                                               Prob > chi2 = 0.0000
Log pseudolikelihood = -10576.261                                Pseudo R2 = 0.0347
```

docvis	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
docvis						
private	.1571795	.0367377	4.28	0.000	.085175	.229184
medicaid	.0860199	.0539415	1.59	0.111	-.0197035	.1917433
age	.30188	.0638297	4.73	0.000	.1767761	.4269838
age2	-.0019838	.0004233	-4.69	0.000	-.0028135	-.0011542
educyr	.0284782	.0049346	5.77	0.000	.0188066	.0381498
actlim	.1875403	.0387357	4.84	0.000	.1116198	.2634607
totchr	.2761519	.0132766	20.80	0.000	.2501303	.3021735
_cons	-10.54756	2.39369	-4.41	0.000	-15.23911	-5.856013
lnalpha						
female	-.1871933	.0732959	-2.55	0.011	-.3308506	-.0435359
bh	.3103148	.0949842	3.27	0.001	.1241493	.4964803
_cons	-.4119142	.0582034	-7.08	0.000	-.5259907	-.2978377

There is some improvement in the log likelihood relative to the NB2 model. The dispersion is greater for blacks and Hispanics and smaller for females. However, these two variables could also have been introduced into the conditional mean function. The decision to let a variable affect α rather than μ can be difficult to justify.

20.3.6 Nonlinear least-squares estimation

Suppose one wants to avoid any parametric specification of the conditional variance function. Instead, one may fit the exponential mean model by nonlinear least squares (NLS) and use a robust estimate of the VCE. For count data, this estimator is likely to be less efficient than the Poisson MLE because the Poisson MLE explicitly models the intrinsic heteroskedasticity of count data, whereas the NLS estimator is based on homoskedastic errors.

The NLS objective function is

$$Q(\boldsymbol{\beta}) = \sum_{i=1}^N \{y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})\}^2$$

Section 13.3.6 provides an NLS application, using the `nls` command, for doctor visits in a related dataset.

A practical complication not mentioned in section 13.3.6 is that if most observations are 0, then the NLS estimator can encounter numerical problems. The NLS estimator can be shown to solve

$$\sum_{i=1}^N \{y_i - \exp(\mathbf{x}'_i \boldsymbol{\beta})\} \exp(\mathbf{x}'_i \boldsymbol{\beta}) \mathbf{x}_i = \mathbf{0}$$

Compared with (20.4) for the Poisson MLE, there is an extra multiple, $\exp(\mathbf{x}'_i \boldsymbol{\beta})$, which can lead to numerical problems if most counts are 0. NLS estimation using the `nls` command yields

```

. * NLS with exponential conditional mean
. nl (docvis = exp({xb: $xlist} + {constant})), vce(robust) nolog

Nonlinear regression                                         Number of obs =      3,677
                                                               R-squared     =     0.5436
                                                               Adj R-squared =     0.5426
                                                               Root MSE      =   6.804007
                                                               Res. dev.     = 24528.25

```

docvis	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]
/xb_private	.1235144	.0395179	3.13	0.002	.0460351 .2009937
/xb_medicaid	.0856747	.0649936	1.32	0.188	-.0417525 .2131018
/xb_age	.2951153	.0720509	4.10	0.000	.1538516 .4363789
/xb_age2	-.0019481	.0004771	-4.08	0.000	-.0028836 -.0010127
/xb_educyr	.0309924	.0051192	6.05	0.000	.0209557 .0410291
/xb_actlim	.1916735	.0413705	4.63	0.000	.110562 .2727851
/xb_totchr	.2191967	.0151021	14.51	0.000	.1895874 .248806
/constant	-10.12438	2.713159	-3.73	0.000	-15.44383 -4.804931

The NLS coefficient estimates are within 20% of the Poisson and NB2 ML estimates, with similar differences for the implied MES. The robust standard errors for the NLS estimates are about 20% higher than those for the Poisson MLE, confirming the expected efficiency loss.

Unless there is good reason to do otherwise, for count data, it is better to use Poisson or NB2 MLES than to use the NLS estimator.

20.3.7 Censored and truncated count regression

Counts are censored if data on the dependent variable are missing for some specific count values, though data on regressors are still available even for observations with missing counts. The leading example is cases where counts are top coded, say, at 10 or more. A second common example is when counts are collected in ranges such as 0, 1, 2, 3–5, 6–10, and more than 10.

The `cpoisson` command provides ML estimates of the Poisson model with left-censoring or right-censoring. A simple example of right-censored (at 10 or more) Poisson regression with regressor `x` is `cpoisson docvis x, ul(10)`. Consistency requires that the underlying complete conditional distribution of `y` is the Poisson. At the time of writing, no similar command is available for the NB.

Counts are truncated if data on the dependent variable are missing for some specific count values, and additionally data on regressors are also unavailable for

those observations. The leading example is left-truncation at zero. For example, only individuals who participate in an activity may be surveyed to find out how many times they participated.

The `tpoisson` command, which supersedes the `ztp` command, provides ML estimates of the Poisson model with left-truncation or right-truncation. And the `tnbreg` command, which supersedes the `ztnb` command, provides ML estimates of the NB2 models with left-truncation. The syntax and options for these commands are the same as those for the `poisson` and `nbreg` commands. In particular, the default for `tnbreg` is to estimate the parameters of a zero-truncated NB2 model; an example with left-truncation is given in section [20.4](#).

20.4 Hurdle model

In this section and the next, we consider two types of mixture models that involve new specifications of both the conditional mean and variance of the distributions.

20.4.1 Hurdle model

The hurdle model, or two-part model, relaxes the assumption that the zeros and the positives come from the same data-generating process. The zeros are determined by the density $f_1(\cdot)$, so that $\Pr(y = 0) = f_1(0)$ and $\Pr(y > 0) = 1 - f_1(0)$. The positive counts come from the truncated density $f_2(y|y > 0) = f_2(y)/\{1 - f_2(0)\}$, which is multiplied by $\Pr(y > 0)$ to ensure that probabilities sum to 1. Thus, suppressing regressors for notational simplicity, we see

$$f(y) = \begin{cases} f_1(0) & \text{if } y = 0 \\ \frac{1 - f_1(0)}{1 - f_2(0)} f_2(y) & \text{if } y \geq 1 \end{cases}$$

This specializes to the standard model only if $f_1(\cdot) = f_2(\cdot)$. Although the motivation for this model is to handle excess zeros, it is also capable of modeling too few zeros.

A hurdle model has the interpretation that it reflects a two-stage decision-making process, each part being a model of one decision. The two parts of the model are functionally independent. Therefore, ML estimation of the hurdle model can be achieved by separately maximizing the two terms in the likelihood, one corresponding to the zeros and the other to the positives. This is straightforward. The first part uses the full sample, but the second part uses only the positive count observations.

For certain types of activities, such a specification is easy to rationalize. For example, in a model that explains the amount of cigarettes smoked per

day, the survey may include both smokers and nonsmokers. One model determines whether one smokes, and a second model determines the number of cigarettes (or packs of cigarettes) smoked given that at least one is smoked.

As an illustration, we obtain draws from a hurdle model as follows. The positives are generated by Poisson(2) truncated at 0. One way to obtain these truncated draws is to draw from Poisson(2) and then replace any zero draw for any observation by a nonzero draw, until all draws are nonzero. This can be shown to be equivalent to the accept–reject method for drawing random variates that is defined in, for example, [Cameron and Trivedi \(2005, 414\)](#). This method is simple but is computationally inefficient if a high fraction of draws are truncated at zero. To then obtain draws from the hurdle model, we randomly replace some of the truncated Poisson draws with zeros. A draw is replaced with a probability of π and kept with a probability $1 - \pi$. We set $\pi = 1 - (1 - e^{-2})/2 \simeq 0.568$ because this can be shown to yield a mean of 1 for the hurdle model draws. The proportion of positives is then 0.432. We have

```

. * Hurdle: Pr(y=0)=pi and Pr(y=k)=(1-pi) x Poisson(2) truncated at 0
. clear
. qui set obs 10000
. set seed 10101          // Set the seed !
. scalar pi=1-(1-exp(-2))/2 // Probability y=0
. generate xhurdle = 0
. scalar minx = 0
. while minx == 0 {
    2. generate xph = rpoisson(2)
    3. qui replace xhurdle = xph if xhurdle==0
    4. drop xph
    5. qui summarize xhurdle
    6. scalar minx = r(min)
    7. }
. replace xhurdle = 0 if runiform() < pi
(5,729 real changes made)
. qui histogram xhurdle, discrete xtitle("Hurdle Poisson")
>     saving(histphurdle, replace)
. summarize xhurdle

```

Variable	Obs	Mean	Std. dev.	Min	Max
xhurdle	10,000	.9891	1.413146	0	8

The setup is such that the random variable has a mean of 1. From the summary statistics, this is the case. The model has induced overdispersion because the variance $1.4131^2 = 1.997 > 1$.

The hurdle model changes the conditional mean specification. Under the hurdle model, the conditional mean is

$$E(y|\mathbf{x}) = \Pr(y_1 > 0|\mathbf{x}_1) \times E_{y_2>0}(y_2|y_2 > 0, \mathbf{x}_2) \quad (20.9)$$

and the two terms on the right are determined by the two respective parts of the model. Because of the form of the conditional mean specification, the calculation of MES, $\partial E(y|\mathbf{x})/\partial x_j$, is more complicated.

20.4.2 Variants of the hurdle model

Any binary outcome model can be used for modeling the zero-versus-positive outcome. Logit is a popular choice. The second part can use any truncated parametric count density, for example, Poisson or NB. In application, the covariates in the hurdle part that models the zero/one outcome need not be the same as those that appear in the truncated part, although in practice they are often the same.

The hurdle model is widely used, and the hurdle NB model is quite flexible. The main drawback is that the model is not very parsimonious. A competitor to the hurdle model is the zero-inflated class of models, presented in section [20.6.2](#).

Two variants of the hurdle count model are provided by the community-contributed `hplogit` and `hnbllogit` commands ([Hilbe and Hardin 2005a,b](#)). They use the logit model for the first part and either the zero-truncated Poisson (ZTP) or the zero-truncated negative binomial (ZTNB) model for the second part. The partial syntax is

```
hplogit depvar [ varlist ] [ if ] [ in ] [ , options ]
```

where *options* include `robust` and `nolog`, as well as many of those for the regression command. Note that these commands restrict the regressors to be the same in each part and do not support factor notation. The `churdle` command fits linear and exponential hurdle models.

20.4.3 Application of the hurdle model

We implement ML estimation of the hurdle model with two-step estimation using official Stata commands, rather than the community-contributed commands, to demonstrate how to proceed if regressors differ in the two parts.

The first step involves estimating the parameters of a binary outcome model, popular choices being binary logit or probit estimated by using `logit` or `probit`. The second step estimates the parameters of a ZTP or ZTNB model, using the `tpoisson` command or the `tnbreg` command.

We first use `logit`. We do not need to transform `docvis` to a binary variable before running the logit because Stata does this automatically. This is easy to verify by doing the transformation and then running the logit.

```
. * Hurdle logit-nb2 model manually: (1) Logit for zeros
. qui use mus220mepsdocvis, clear
. logit docvis $xlist, nolog vce(robust)

Logistic regression                                         Number of obs = 3,677
                                                               Wald chi2(7) = 281.39
                                                               Prob > chi2 = 0.0000
Log pseudolikelihood = -1040.3258                           Pseudo R2 = 0.1788
```

docvis	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
private	.6586978	.1292037	5.10	0.000	.4054631	.9119324
medicaid	.0554225	.1738421	0.32	0.750	-.2853017	.3961468
age	.5428779	.2290737	2.37	0.018	.0939016	.9918542
age2	-.0034989	.0015365	-2.28	0.023	-.0065104	-.0004873
educyr	.047035	.0153971	3.05	0.002	.0168571	.0772128
actlim	.1623927	.1540586	1.05	0.292	-.1395567	.4643421
totchr	1.050562	.0703947	14.92	0.000	.9125907	1.188533
_cons	-20.94163	8.49238	-2.47	0.014	-37.58638	-4.296868

The second-step regression is based only on the sample with positive observations for `docvis`.

```
. * Hurdle logit-nb2 model manually: (2a) restrict to positives only
. summarize docvis if docvis > 0
```

Variable	Obs	Mean	Std. dev.	Min	Max
docvis	3,276	7.657814	7.415095	1	144

Dropping zeros from the sample has raised the mean and lowered the standard deviation of `docvis`.

The parameters of a ZTNB model are then estimated next by using `tnbreg`.

```

. * Hurdle logit-nb2 model manually: (2b) ZTNB for positives
. tnbreg docvis $xlist if docvis>0, nolog vce(robust)

Truncated negative binomial regression
Truncation point = 0
Dispersion: mean
Log pseudolikelihood = -9452.899

Number of obs = 3,276
Wald chi2(7) = 474.34
Prob > chi2 = 0.0000
Pseudo R2 = 0.0262

```

docvis	Robust					[95% conf. interval]
	Coefficient	std. err.	z	P> z		
private	.1095567	.0382086	2.87	0.004	.0346692	.1844442
medicaid	.0972309	.0589629	1.65	0.099	-.0183342	.212796
age	.2719032	.0671328	4.05	0.000	.1403254	.403481
age2	-.0017959	.000445	-4.04	0.000	-.002668	-.0009238
educyr	.0265974	.0050938	5.22	0.000	.0166137	.0365812
actlim	.1955384	.040658	4.81	0.000	.1158503	.2752266
totchr	.2226969	.0135761	16.40	0.000	.1960882	.2493056
_cons	-9.19017	2.517163	-3.65	0.000	-14.12372	-4.256621
/lnalpha	-.5259629	.0544868			-.632755	-.4191708
alpha	.590986	.0322009			.5311265	.6575919

A positively signed coefficient in the logit model means that the corresponding regressor increases the probability of a positive observation. In the second part, a positive coefficient means that, conditional on a positive count, the corresponding variable increases the value of the count. The results show that all the variables except `medicaid` and `actlim` have statistically significant coefficients and that they affect both the outcomes in the same direction.

For this example with a common set of regressors in both parts of the model, the community-contributed `hnblogit` command can instead be used. Then,

```

. * Same hurdle model fit using the community-contributed hnblogit command
. hnblogit docvis $xlist, vce(robust)
    (output omitted)

```

yields the same parameter estimates as the separate estimation of the two components of the model.

It is straightforward to calculate MES for the two components separately. To ensure that MES for binary regressors are calculated using the finite-

difference method, we need to use the factor-variable `i.` operator in defining the model to be fit. And when we replace `age` and `age2` in the regressor list with `c.age` and `c.age#c.age`, the `margins` command will give the correct ME with respect to `age`; see section 13.7.11.

The sample AMES for whether a doctor visit occurs is obtained by using `margins, dydx(*)` after `logit`.

```
. * Hurdle logit-nb2: AMEs for first part
. global xlistfv i.private i.medicaid c.age##c.age educyr i.actlim totchr
. qui logit docvis $xlistfv, vce(robust)
. margins, dydx(*)

Average marginal effects                                         Number of obs = 3,677
Model VCE: Robust

Expression: Pr(docvis), predict()
dy/dx wrt: 1.private 1.medicaid age educyr 1.actlim totchr
```

	Delta-method				
	dy/dx	std. err.	z	P> z	[95% conf. interval]
1.private	.0552042	.0106987	5.16	0.000	.0342351 .0761733
1.medicaid	.0046029	.0142787	0.32	0.747	-.0233829 .0325886
age	.0024871	.0009149	2.72	0.007	.0006939 .0042803
educyr	.0039512	.0012887	3.07	0.002	.0014255 .0064769
1.actlim	.0133166	.0122861	1.08	0.278	-.0107637 .037397
totchr	.088253	.0055746	15.83	0.000	.0773269 .0991791

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

Having supplemental private insurance and an additional chronic condition are big determinants of whether a doctor visit occurs.

The sample AMES for the second truncated part are obtained by using

```

. * Hurdle logit-nb2: AMEs for second part
. qui tnbgreg docvis $xlistfv if docvis>0, vce(robust)
. margins, dydx(*)

Average marginal effects                                         Number of obs = 3,276
Model VCE: Robust

Expression: Predicted number of events, predict()
dy/dx wrt: 1.private 1.medicaid age educyr 1.actlim totchr

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.private	.7878003	.2745538	2.87	0.004	.2496848	1.325916
1.medicaid	.7220008	.453203	1.59	0.111	-.1662607	1.610262
age	.0280808	.0186977	1.50	0.133	-.008566	.0647277
educyr	.191417	.0371488	5.15	0.000	.1186067	.2642273
1.actlim	1.432509	.3066782	4.67	0.000	.831431	2.033587
totchr	1.60271	.1039497	15.42	0.000	1.398972	1.806448

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

Medicaid and having an activity limitation are now bigger determinants than in the first part of the model.

Ideally, we additionally compute the ME with respect to the conditional mean given in (20.9) because change in a regressor may change both the logit and the truncated count components of the model. This is possible for the Poisson-logit model using the `gsem` command presented in section 23.6, followed by a more complicated application of the `margins, dydx(*)` command that uses the `expression()` option.

```

. * Hurdle logit-poisson: AMEs for combined model
. generate visit = (docvis > 0)
. generate novisit = 1 - visit
. qui gsem (docvis <- $xlistfv, family(poisson, ltruncated(novisit)))
>     (visit <- $xlistfv, logit), vce(robust)
. margins, dydx(*) expression( (exp(predict(eta))/(1-exp(-exp(predict(eta))))) *
>     * predict(equation(visit)) )
Average marginal effects                                         Number of obs = 3,677
Model VCE: Robust
Expression: (exp(predict(eta))/(1-exp(-exp(predict(eta))))) *
predict(equation(visit))
dy/dx wrt: 1.private 1.medicaid age educyr 1.actlim totchr

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.private	.9540639	.2448103	3.90	0.000	.4742445	1.433883
1.medicaid	.5987787	.4099716	1.46	0.144	-.2047509	1.402308
age	.0352164	.0170652	2.06	0.039	.0017693	.0686635
educyr	.1988132	.0333316	5.96	0.000	.1334844	.264142
1.actlim	1.341224	.2841913	4.72	0.000	.7842192	1.898229
totchr	1.8226	.0889377	20.49	0.000	1.648285	1.996915

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

The first part of the `gsem` command fits the logit model, and the second part fits the truncated Poisson with exponential conditional mean as a special case of a left-truncated generalized linear model. The `expression()` option of the `margins` command defines the quantity of interest, here the conditional mean given in (20.9). The term $\Pr(y_1 > 0 | \mathbf{x}_1) = \Lambda(\mathbf{x}'_1 \boldsymbol{\beta}_1)$ is computed by `predict(equation(visit))` because the default prediction for the logit model for `visit` is the fitted probability that `visit = 1`. For the truncated Poisson $E_{y_2 > 0}(y_2 | y_2 > 0, \mathbf{x}_2) = e^{\mathbf{x}'_2 \boldsymbol{\beta}_2} / \{1 - \exp(-e^{\mathbf{x}'_2 \boldsymbol{\beta}_2})\}$ and for a generalized linear model, the predicted value of $\mathbf{x}'_2 \boldsymbol{\beta}_2$ is stored in `eta`.

The AMES for the Poisson-logit hurdle model are within 20% of those for the simpler Poisson model, a result that might be expected because only 11% of the sample has zero doctor visits.

The `gsem` command cannot fit an NB-logit model. In this particular example, with relatively few observations with zero doctor visits, the ZTP regression estimates are similar to the ZTNB estimates, so the ME for the

conditional mean ([20.9](#)) will be quite similar. If interest lies in predicting probabilities, however, then the NB-logit model should be used.

The discussion of model selection is postponed to later in this chapter.

20.5 Finite-mixture models

The Poisson distribution is the natural starting point for modeling count data because from stochastic process theory, a pure Poisson point process generates an exponential distribution for the interval between occurrences of an event and a Poisson distribution for the number of events in a given interval. Poisson regression models introduce regressors to allow different individuals to come from Poisson processes with different means, but in practice there is still unobserved heterogeneity that leads to microeconometrics data typically not being equidispersed, even after inclusion of regressors.

Richer parametric models for counts therefore explicitly introduce unobserved heterogeneity. The NB model introduces a heterogeneity variable, ν , that is assumed to have a continuous distribution (gamma). This is an example of a continuous mixture model.

An alternative approach instead uses a discrete representation of unobserved heterogeneity. This generates a class of models called finite-mixture models (FMMs)—a particular subclass of latent-class models; see [Deb \(2007\)](#) and [Cameron and Trivedi \(2005, 678–679\)](#).

20.5.1 FMM specification

An FMM specifies that the density of y is a linear combination of m different densities, where the j th density is $f_j(y|\beta_j)$, $j = 1, 2, \dots, m$. Thus, an m -component finite mixture is

$$f(y|\beta, \pi) = \sum_{j=1}^m \pi_j f_j(y|\beta_j), \quad 0 \leq \pi_j \leq 1, \quad \sum_{j=1}^m \pi_j = 1$$

A simple example is a two-component ($m = 2$) Poisson mixture of $\text{Poisson}(\mu_1)$ and $\text{Poisson}(\mu_2)$. This may reflect the possibility that the sampled population contains two “types” of cases, whose y outcomes are

characterized by the distributions $f_1(y|\beta_1)$ and $f_2(y|\beta_2)$, which are assumed to have different moments. The mixing fraction, π_1 , is in general an unknown parameter. In a more general formulation, it, too, can be parameterized for the observed variables z .

20.5.2 Simulated finite-mixture sample with comparisons

As an illustration, we generate a mixture of Poisson(0.5) and Poisson(5.5) in proportions 0.9 and 0.1, respectively.

```
. * Finite Mixture: Poisson(.5) with prob .9 and Poisson(5.5) with prob .1
. set seed 10101          // Set the seed !
```

```
. generate xp1= rpoisson(.5)
. generate xp2= rpoisson(5.5)
. summarize xp1 xp2
```

Variable	Obs	Mean	Std. dev.	Min	Max
xp1	10,000	.5007	.7041655	0	5
xp2	10,000	5.4776	2.332433	0	17

```
. rename xp1 xpmix
. qui replace xpmix = xp2 if runiform() > 0.9
. qui histogram xpmix, discrete xtitle("Finite-mixture Poisson")
>      saving(histp2mix, replace)
. summarize xpmix
```

Variable	Obs	Mean	Std. dev.	Min	Max
xpmix	10,000	.9696	1.730831	0	14

The setup yields a random variable with a mean of $0.9 \times 0.5 + 0.1 \times 5.5 = 1$. But the data are overdispersed, with a variance in this sample of $1.731^2 = 2.996$. This dispersion is greater than those for the preceding generated data samples from Poisson, NB2, and hurdle models.

```
. * Finite Mixture: Relative frequencies
. tabulate xpmix
```

xpmix	Freq.	Percent	Cum.
0	5,459	54.59	54.59
1	2,794	27.94	82.53
2	725	7.25	89.78
3	253	2.53	92.31
4	166	1.66	93.97
5	188	1.88	95.85
6	157	1.57	97.42
7	89	0.89	98.31
8	79	0.79	99.10
9	42	0.42	99.52
10	27	0.27	99.79
11	14	0.14	99.93
12	4	0.04	99.97
14	3	0.03	100.00
Total	10,000	100.00	

As for the NB2, the distribution has a long right tail. Although the component means are far apart, the mixture distribution is not bimodal; see the histogram in figure [20.1](#). This is because only 10% of the observations come from the high-mean distribution.

It is helpful to view graphically the four distributions generated in this chapter: Poisson, NB2, hurdle, and finite mixture. All have the same mean of 1, but they have different dispersion properties. The generated data were used to produce four histograms that we now combine into a single graph.

```
. * Histograms of four different distributions, all with mean 1
. graph combine histpois.gph histnb2.gph histphurdle.gph histp2mix.gph,
>    iscale(1.1) ycommon xcommon
>    title("Four different distributions with mean = 1")
```

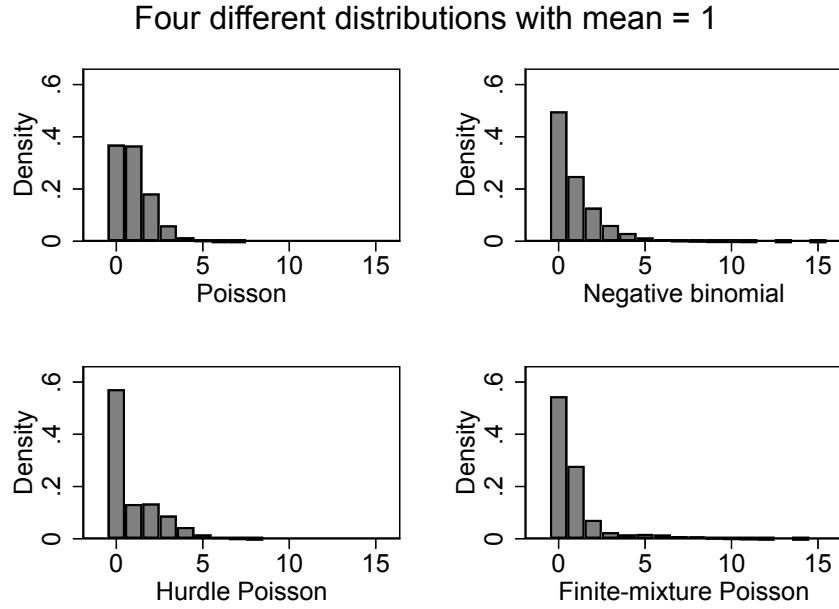


Figure 20.1. Four count distributions with the same mean

It is helpful for interpretation to supplement this graph with summary statistics for the distributions:

- . * Means and standard deviations of four different distributions, all with mean 1
- . summarize xpois xnegbin xhurdle xpmix

Variable	Obs	Mean	Std. dev.	Min	Max
xpois	10,000	.9989	1.004689	0	7
xnegrbin	10,000	.9982	1.410814	0	12
xhurdle	10,000	.9891	1.413146	0	8
xpmix	10,000	.9696	1.730831	0	14

20.5.3 ML estimation of the FMM

The components of the mixture may be assumed, for generality, to differ in all their parameters. This is a more flexible specification because all moments of the distribution depend upon $(\pi_j, \beta_j, j = 1, \dots, m)$. But such flexibility comes at the expense of parsimonious parameterization. More parsimonious formulations assume that only some parameters differ across the components, for example, the intercepts, and the remaining parameters are common to the mixture components.

ML estimation of an FMM is computationally challenging because the log-likelihood function may be multimodal and not logconcave and because individual components may be hard to identify empirically. If numerical problems are encountered for a three-component model, for example, then a two-component model might be used to provide initial values for the estimation of the three-component model. The presence of outliers in the sample may cause further identification problems.

The `fmm` prefix

The `fmm` prefix enables ML estimation of finite-mixture count models. The command can be used to estimate mixtures of several continuous and count models. Section 14.2 covered the application of the `fmm` prefix to the normal linear regression model. In this chapter, we cover regression models for counts.

The syntax for the standard version of this command is as follows,

```
fmm # [if] [in] [weight] [, fmmopts]: model depvar indepvars [, options]
```

where `#` refers to the number of components in the specification and *model* refers to the specification of the distribution.

The syntax for the hybrid version of the model, which we explain and illustrate in section [20.5.8](#), is as follows:

```
fmm [if] [in] [weight] [, fmmopts]:  
(model depvar indepvars [, lcprob(varlist) options]) (component_2) ...
```

The default setup assumes class probabilities do not vary over individuals. The command supports factor-variable operators and the `vce()` option with all the usual types of VCE.

An important option is `lcprob(varlist)`, which allows the π_j to be parameterized as a function of the variables in *varlist*. A multinomial logit model is specified for the latent class probabilities, with normalization on the first component, so

$$\pi_j = \frac{\exp(\mathbf{z}'_j \boldsymbol{\beta})}{1 + \sum_{k=2}^m \exp(\mathbf{z}'_k \boldsymbol{\beta})}$$

20.5.4 Application: Poisson FMM

We apply the `fmm` prefix to the doctor-visit data, using both Poisson and NB2 variants.

In a two-component Poisson mixture, denoted by FMM2-P, each component is a Poisson distribution with a different mean, that is, $\text{Poisson}\{\exp(\mathbf{x}'\boldsymbol{\beta}_j)\}$, $j = 1, 2$, and the proportion π_j of the sample comes from each subpopulation. This model will have $2K + 1$ unknown parameters, where K is the number of exogenous variables in the model. For the two-component NB mixture, denoted by FMM2-NB, a similar interpretation applies, but now the overdispersion parameters also vary between subpopulations. This model has $2(K + 1) + 1$ unknown parameters.

We first fit the FMM2-P model.

```
. * FMM two-component Poisson with constant probabilities
. qui use mus220mepsdocvis, clear
. global xlist i.private i.medicaid c.age##c.age c.educyr i.actlim c.totchr
. fmm 2, nolog vce(robust): poisson docvis $xlist
Finite mixture model                                         Number of obs = 3,677
Log pseudolikelihood = -12100.185
```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1.Class	(base outcome)					
2.Class	_cons	-.5980831	.1171272	-5.11	0.000	-.8276481 -.368518

Class: 1
 Response: docvis
 Model: poisson

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
docvis					
1.private	.2393558	.0695013	3.44	0.001	.1031357 .3755759
1.medicaid	.0463821	.0884509	0.52	0.600	-.1269785 .2197427
age	-.6233526	.1367728	-4.56	0.000	-.8914223 -.3552829
c.age#c.age	.0045366	.0009492	4.78	0.000	.0026762 .0063971
educyr	.0284599	.0078417	3.63	0.000	.0130905 .0438294
1.actlim	.1723268	.0733314	2.35	0.019	.0285999 .3160537
totchr	.3286694	.0215553	15.25	0.000	.2864218 .3709169
_cons	21.35464	4.881451	4.37	0.000	11.78717 30.92211

Class: 2
 Response: docvis
 Model: poisson

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
docvis					
1.private	.1566873	.0656687	2.39	0.017	.0279789 .2853957
1.medicaid	.1924436	.1383488	1.39	0.164	-.078715 .4636022
age	1.232368	.112787	10.93	0.000	1.011309 1.453426
c.age#c.age	-.0085471	.0007425	-11.51	0.000	-.0100024 -.0070917
educyr	.0219929	.0084774	2.59	0.009	.0053775 .0386082
1.actlim	.1486859	.0825088	1.80	0.072	-.0130284 .3104003
totchr	.1898829	.03185	5.96	0.000	.127458 .2523078
_cons	-42.46506	4.251345	-9.99	0.000	-50.79755 -34.13258

The computer output begins with estimates of the mixture probabilities, fit as a logit model with normalization on the first class. In this example, the probabilities do not depend on regressors, so for all observations we obtain that the probability of being in the second latent class or component is $(1 - \pi) = \exp(-0.5981)/\{1 + \exp(-0.5981)\} = 0.3548$.

The computer output separates the parameter estimates for the two components. If the two latent classes differ a lot in their responses to the changes in the regressors, we would expect the parameters to also differ. In

this example, the differentiation does not appear to be very sharp at the level of individual slope coefficients. But as we see below, this is misleading because the two components have substantially different mean numbers of doctor visits, due perhaps to different intercept estimates or the different quadratics in age. This leads to quite different MES even though the slope parameters do not seem to be all that different.

Distribution of fitted means for each component

These classes are latent, so it is helpful to give them some interpretation.

One natural interpretation is that classes differ in terms of the mean of their respective distributions; that is, $\exp(\mathbf{x}'_i \boldsymbol{\beta}_1) \neq \exp(\mathbf{x}'_i \boldsymbol{\beta}_2)$. To make this comparison, we generate fitted values by using the `predict` command. For the Poisson model, the predictions are $\hat{y}_i^j = \exp(\mathbf{x}'_i \hat{\boldsymbol{\beta}}_j)$, $j = 1, 2$.

The `predict` postestimation command generates predictions for each observation of the two components that are stored as `yhatp1` and `yhatp2`.

```
. * FMM two-component Poisson: Predicted y's and histogram for each component
. qui fmm 2, vce(robust): poisson docvis $xlist
. predict yhatp*
(option mu assumed)
. summarize yhatp1 yhatp2
```

Variable	Obs	Mean	Std. dev.	Min	Max
yhatp1	3,677	5.050474	4.269029	.9507732	50.52482
yhatp2	3,677	11.65096	5.670928	.6697477	58.63944

```
. qui histogram yhatp1, name(class_1, replace)
. qui histogram yhatp2, name(class_2, replace)
. qui graph combine class_1 class_2, iscale(1.2) rows(1) ycommon xcommon
>      ysize(2.5) xsize(6)
```

The summary statistics make explicit the implication of the mixture model. The first component has a relatively low mean number of doctor visits, around 5.05. The second component has a relatively high mean number of doctor visits, around 11.65.

Figure 20.2 provides histograms for each component. Clearly, the second component experiences more doctor visits.

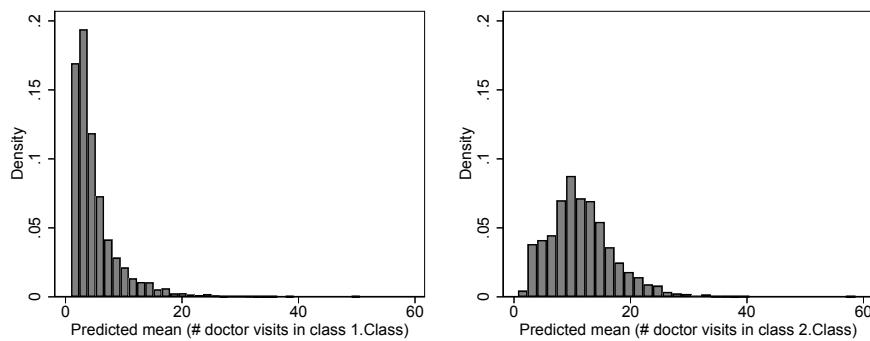


Figure 20.2. Fitted values distribution, FMM2-P

Marginal predicted means for each component

The `estat lcmean` postestimation command gives the marginal predicted means for each component or latent class.

```
. * FMM two-component Poisson: Compute the component marginal predicted means
. estat lcmean
Latent class marginal means                                         Number of obs = 3,677
```

	Delta-method					[95% conf. interval]
	Margin	std. err.	z	P> z		
1 docvis	5.050474	.2713084	18.62	0.000	4.518719	5.582229
2 docvis	11.65096	.6255662	18.62	0.000	10.42488	12.87705

The marginal predicted means are simply the average of the fitted means $\exp(\mathbf{x}'_i \hat{\beta}_j)$ in each component. The `estat lcmean` output controls for estimator imprecision and includes standard errors and confidence intervals.

Marginal predicted probabilities for each component

The `estat lcprob` postestimation command gives the marginal predicted probabilities for each component or latent class. In this example, the probabilities do not vary over individuals, and from the initial output, we computed that the probability for the second component was 0.3548.

```
. * FMM two-component Poisson: Compute the component mean predicted probabilities
. estat lcprob
```

Latent class marginal probabilities Number of obs = 3,677

	Delta-method			
	Margin	std. err.	[95% conf. interval]	
Class				
1	.6452176	.0268118	.5911008	.6958574
2	.3547824	.0268118	.3041426	.4088992

The mixing proportions π_1 and $1 - \pi_1$ equal 0.65 and 0.35. The probability-weighted average number of doctor visits over the two classes is $0.6452 \times 5.0505 + 0.3548 \times 11.6510 = 7.39$, which is close to the overall sample average of 6.82.

In summary, the FMM has the interpretation that the data are generated by two classes of individuals, the first of which accounts for about 65% of the population who are relatively low users of doctor visits and the second of which accounts for about 35% of the population who are high users of doctor visits.

ME

The `margins, dydx(*)` command gives the AME for each regressor, averaged across the two components.

We have

```

. * FMM two-component Poisson: MEs
. margins, dydx(*)

Average marginal effects                                         Number of obs = 3,677
Model VCE: Robust

Expression: Predicted mean (# doctor visits), using class probabilities,
            predict(mu outcome(docvis))

dy/dx wrt: 1.private 1.medicaid age educyr 1.actlim totchr

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.private	1.428504	.3638522	3.93	0.000	.7153672	2.141642
1.medicaid	1.001208	.8365001	1.20	0.231	-.6383022	2.640718
age	.1974592	.0472302	4.18	0.000	.1048897	.2900288
educyr	.1836498	.0502039	3.66	0.000	.085252	.2820477
1.actlim	1.192519	.4101055	2.91	0.004	.3887271	1.996311
totchr	1.855912	.1470048	12.62	0.000	1.567788	2.144036

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

MES for each component can be obtained using the `class()` option. For example, the command `margins, dydx(*) predict(mu class(1)) predict(mu class(2))` yields AMES that differ considerably across the two classes.

20.5.5 Application: NB2 FMM

The `fmm` prefix with the `nbreg` model can be used to estimate a mixture distribution with NB2 components.

This model involves additional overdispersion parameters that can potentially create problems for convergence of the numerical algorithm. This may happen if an overdispersion parameter is too close to zero. Further, the number of parameters increases linearly with the number of components, and the likelihood function quickly becomes high dimensional when the specification includes many regressors.

The two-component NB2 FMM estimation yields

```

. * FMM two-component negative binomial with constant probabilities
. fmm 2, nolog vce(robust): nbreg docvis $xlist
Finite mixture model                                         Number of obs = 3,677
Log pseudolikelihood = -10534.237

```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1.Class	(base outcome)					
2.Class						
_cons	.2286522	.6670507	0.34	0.732	-1.078743	1.536048

Class: 1
 Response: docvis
 Model: nbreg, dispersion(mean)

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
docvis						
1.private	.3292254	.1225044	2.69	0.007	.0891212	.5693296
1.medicaid	.1319335	.14372	0.92	0.359	-.1497525	.4136196
age	.3818359	.1483224	2.57	0.010	.0911293	.6725425
c.age#c.age	-.0024401	.0009896	-2.47	0.014	-.0043796	-.0005005
educyr	.0383352	.0119704	3.20	0.001	.0148738	.0617967
1.actlim	.066328	.1344501	0.49	0.622	-.1971894	.3298454
totchr	.5013939	.0755416	6.64	0.000	.353335	.6494528
_cons	-15.15966	5.686649	-2.67	0.008	-26.30529	-4.014036
/docvis						
lnalpha	-.9786601	.4212083			-1.804213	-.153107

Class: 2
 Response: docvis
 Model: nbreg, dispersion(mean)

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
docvis						
1.private	.0957578	.0516769	1.85	0.064	-.0055271	.1970427
1.medicaid	.0901739	.0891725	1.01	0.312	-.0846011	.2649488
age	.2691574	.0890089	3.02	0.002	.0947032	.4436116
c.age#c.age	-.0017901	.0005887	-3.04	0.002	-.002944	-.0006363
educyr	.0241906	.0078689	3.07	0.002	.0087679	.0396133
1.actlim	.219569	.0550544	3.99	0.000	.1116643	.3274738
totchr	.1758912	.0560492	3.14	0.002	.0660368	.2857456
_cons	-8.645314	3.41952	-2.53	0.011	-15.34745	-1.943178
/docvis						
lnalpha	-.7697485	.093542			-.9530873	-.5864096

The maximized value of the log likelihood ($-10,534$) is much higher than that for the two-component Poisson ($-12,100$) (section [20.5.4](#)). It is lower, however, than that for the hurdle NB2 ($-10,493$) (section [20.4.3](#)), even though there are three more parameters in the mixture model. As expected, there is evidence of overdispersion in both components; see `lnalpha`.

A comparison of the means of the probabilities and predicted means of the two components is possible by using the `estat lcprob` and `estat lcmean` commands.

```
. * FMM two-component NB2: Component marginal predicted probabilities and means
. estat lcprob
```

Latent class marginal probabilities Number of obs = 3,677

		Delta-method		
		Margin	std. err.	[95% conf. interval]
Class	1	.4430847	.1646019	.1771106 .7462561
	2	.5569153	.1646019	.2537439 .8228894

```
. estat lcmean
```

Latent class marginal means Number of obs = 3,677

		Delta-method				
		Margin	std. err.	z	P> z	[95% conf. interval]
1	docvis	4.477236	1.053625	4.25	0.000	2.412169 6.542303
2	docvis	8.828413	.6283404	14.05	0.000	7.596889 10.05994

The two classes are somewhat different in probability of occurrence. The first class accounts for about 44% of the population who are relatively low users of doctor visits (average visits 4.5), and the second class accounts for about 56% of the population who are high users (average visits 8.8).

After the `fmm` prefix is executed, we can run the `predict` postestimation command to generate component class-specific predicted conditional means for each observation. Histograms then provide useful graphical summaries of the spread, overlap, and separation between component distributions. This analysis was presented for the FMM Poisson example, so it is not repeated here.

20.5.6 Latent class posterior probabilities

The finite mixture model can be interpreted as a latent class model. In this interpretation, an individual comes from exactly one of m classes.

The probability of individual i coming from class j is then not simply that individual mixing probability π_{ji} . Instead, the mixing probability is

weighted by the density $f_j(y_{ji}|\beta_j)$. The posterior probability for each observation, the estimated probability that an observation belongs to a particular latent class, is

$$\tilde{\pi}_{ji} = \frac{\pi_{ji} f_j(y_i|\beta_j)}{\sum_{k=1}^m \pi_{ki} f_k(y_i|\beta_k)}$$

Note that the posterior probabilities vary with individual characteristics, even if the mixing probabilities do not.

This calculation is done using the `predict, classposterior` postestimation command.

The following example computes the posterior probabilities of being in each class for the two-component Poisson and NB2 models.

```
. * Posterior probabilities of each latent class for Poisson and NB2
. qui fmm 2, vce(robust): poisson docvis $xlist
. predict postpois*, classposteriorpr
. qui fmm 2, vce(robust): nbreg docvis $xlist
. predict postnb*, classposteriorpr
. sum postpois* postnb*
```

Variable	Obs	Mean	Std. dev.	Min	Max
postpois1	3,677	.6452176	.4104135	0	1
postpois2	3,677	.3547824	.4104135	8.23e-19	1
postnb1	3,677	.4430855	.2403444	1.03e-25	.8714064
postnb2	3,677	.5569145	.2403444	.1285937	1

For the Poisson FMM, the posterior probabilities of being in the first latent class range from 0.0 to 1.0, whereas the constant mixing probability $\hat{\pi}_1 = 0.6452$ for all observations. The average of the 3,677 posterior probabilities does equal $\hat{\pi}_1 = 0.6452$. For NB2 FMM, the posterior probabilities of being in the first class range from 0 to 0.8714.

Kernel density graphs of the latent class posterior probabilities provide useful visual information about the separation between the two classes.

```

. * Kernel densities for posterior probabilities of each class for Poisson and NB2
. kdensity postpois1, lwidth(medthick) title(" ") name(postpois1, replace)
. kdensity postpois2, lwidth(medthick) title(" ") name(postpois2, replace)
. kdensity postnb1, lwidth(medthick) title(" ") name(postnb1, replace)
. kdensity postnb2, lwidth(medthick) title(" ") name(postnb2, replace)
. graph combine postpois1 postpois2 postnb1 postnb2, rows(2) ycommon
>      xcommon ysize(5) xsize(6) iscale(0.8)

```

Figure 20.3 provides histograms for the posterior probabilities of being in each latent class for, respectively, the two-component Poisson and NB2 models. The smaller the overlap between the densities, the better the separation of the distributions.

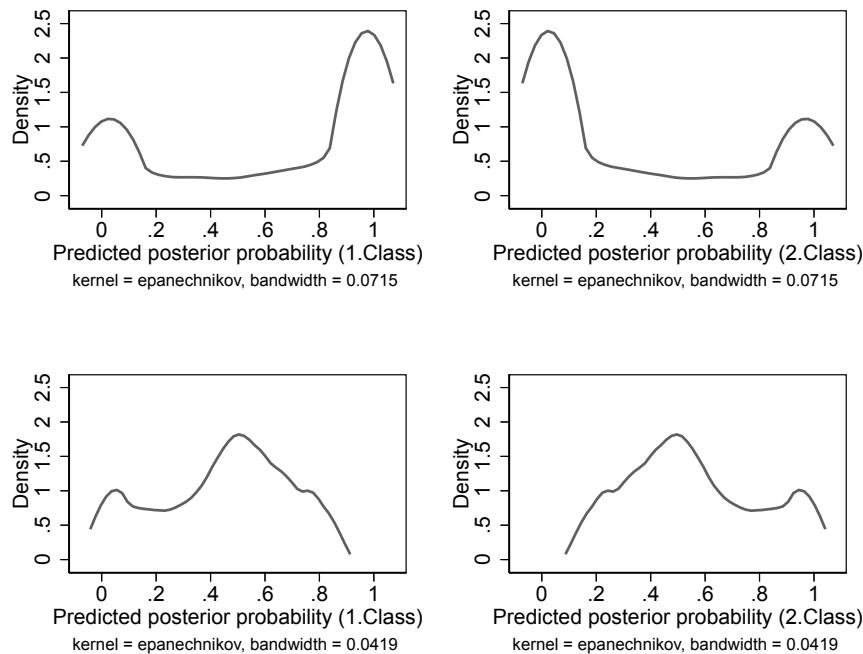


Figure 20.3. Densities of posterior probabilities for FMM Poisson and NB2

20.5.7 Testing the equality of coefficients of mixture components

We can formally test the equality of coefficients in the two mixture components. This test is potentially helpful in interpreting the results in

terms of regressors that contribute significantly to the separation of latent classes.

To run the test, we first replay the results of the `fmm` command with the `coeflegend` option, which provides the list of Stata's internal reference names of the coefficients.

```
. * Use coeflegend to find complete names of model coefficients
. qui fmm 2, vce(robust): poisson docvis $xlist
. fmm, coeflegend
```

Finite mixture model
Log pseudolikelihood = -12100.185

Number of obs = 3,677

	Coefficient Legend
1.Class	(base outcome)
2.Class _cons	-.5980831 _b[2.Class:_cons]

Class: 1
Response: docvis
Model: poisson

	Coefficient Legend
docvis	
1.private	.2393558 _b[docvis:1.private#1.Class]
1.medicaid	.0463821 _b[docvis:1.medicaid#1.Class]
age	-.6233526 _b[docvis:1.Class#c.age]
c.age#c.age	.0045366 _b[docvis:1.Class#c.age#c.age]
educyr	.0284599 _b[docvis:1.Class#c.educyr]
1.actlim	.1723268 _b[docvis:1.actlim#1.Class]
totchr	.3286694 _b[docvis:1.Class#c.totchr]
_cons	21.35464 _b[docvis:1.Class]

Class: 2
Response: docvis
Model: poisson

	Coefficient Legend
docvis	
1.private	.1566873 _b[docvis:1.private#2.Class]
1.medicaid	.1924436 _b[docvis:1.medicaid#2.Class]
age	1.232368 _b[docvis:2.Class#c.age]
c.age#c.age	-.0085471 _b[docvis:2.Class#c.age#c.age]
educyr	.0219929 _b[docvis:2.Class#c.educyr]
1.actlim	.1486859 _b[docvis:1.actlim#2.Class]
totchr	.1898829 _b[docvis:2.Class#c.totchr]
_cons	-42.46506 _b[docvis:2.Class]

We consider a test of whether the coefficient of `totchr` is the same across the two components, using the `test` command and using the `contrast` command.

```
. test (_b[docvis:1.Class#c.totchr] = _b[docvis:2.Class#c.totchr])
( 1) [docvis]1bn.Class#c.totchr - [docvis]2.Class#c.totchr = 0
      chi2( 1) =    12.10
      Prob > chi2 =    0.0005
. contrast c.totchr#a.Class, equation(docvis)
Contrasts of marginal linear predictions
Margins: asbalanced
```

	df	chi2	P>chi2
docvis Class#c.totchr	1	12.10	0.0005

	Contrast	Std. err.	[95% conf. interval]
docvis Class#c.totchr (1 vs 2)	.1387865	.039905	.0605742 .2169988

The two different commands lead to identical outcomes. The null hypothesis of equality of coefficients is easily rejected.

20.5.8 Mixtures with varying mixing probability

We next illustrate how additional flexibility of the mixture specification can be obtained by specifying the mixing proportion as a logit model that necessarily restricts the component probabilities to the $[0, 1]$ interval. In general, however, it is difficult to separate the effect of a regressor on the latent class mean from that on the latent class probability, and it is common to specify constant probabilities as in the examples to date.

Here we use the `lcprob()` option to allow the component probabilities to vary with `actlim` and `totchr`. In our specification, these variables are then excluded from the conditional mean function; this restriction is not essential.

```

. * FMM two-component Poisson: lcprob() option to allow varying mixture
> proportions
. fmm 2, lcprob(actlim totchr) nolog vce(robust): poisson docvis private
> medicaid age age2 educyr
Finite mixture model                                         Number of obs = 3,677
Log pseudolikelihood = -12037.83

```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1.Class	(base outcome)					
2.Class						
actlim	.3811733	.1052316	3.62	0.000	.1749231	.5874235
totchr	.6328327	.0418741	15.11	0.000	.550761	.7149043
_cons	-2.23104	.1079409	-20.67	0.000	-2.4426	-2.019479

Class: 1
 Response: docvis
 Model: poisson

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
docvis						
private	.2326691	.057033	4.08	0.000	.1208865	.3444517
medicaid	.323233	.0993596	3.25	0.001	.1284916	.5179743
age	.3376277	.0953861	3.54	0.000	.1506745	.5245809
age2	-.002163	.0006333	-3.42	0.001	-.0034042	-.0009219
educyr	.0412185	.007761	5.31	0.000	.0260073	.0564297
_cons	-12.49424	3.566101	-3.50	0.000	-19.48367	-5.504816

Class: 2
 Response: docvis
 Model: poisson

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
docvis						
private	.1214211	.0611611	1.99	0.047	.0015476	.2412946
medicaid	.2310901	.1269448	1.82	0.069	-.017717	.4798973
age	.1773114	.1115603	1.59	0.112	-.0413426	.3959655
age2	-.0011763	.0007359	-1.60	0.110	-.0026186	.0002661
educyr	.0291441	.0077511	3.76	0.000	.0139523	.044336
_cons	-4.383622	4.194363	-1.05	0.296	-12.60442	3.837178

The first set of output gives estimates for a logit model of the probability of the second component. Given the positive coefficients, having an activity limitation and more chronic conditions will push an individual into the second latent class, which on average has a higher number of doctor visits. The maximized likelihood for this specification (-12,038) is higher than for the model with constant latent class probability (-12,100).

The postestimation summary statistics are as follows:

```
. * FMM two-component Poisson: Postestimation summary statistics
. estat lcmean
```

Latent class marginal means						Number of obs = 3,677
	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
1	docvis	3.374006	.1267247	26.62	0.000	3.12563 3.622381
2	docvis	14.6311	.5826224	25.11	0.000	13.48918 15.77301

```
. estat lcprob
```

Latent class marginal probabilities						Number of obs = 3,677
	Delta-method					
	Margin	std. err.	[95% conf. interval]			
Class						
1	.6918441	.0177926	.6559241	.725583		
2	.3081559	.0177926	.274417	.3440759		

```
. estimates stats
```

Akaike's information criterion and Bayesian information criterion

Model	N	ll(null)	ll(model)	df	AIC	BIC
.	3,677	.	-12037.83	15	24105.66	24198.81

Note: BIC uses N = number of observations. See [R] BIC note.

The latent class marginal means for class 1 and class 2 are, respectively, 3.37 and 14.63 visits. The latent class probabilities are probabilities from the logit model averaged over the 3,677 individuals.

20.5.9 Mixtures with different sets of regressors in each component

The specification of the finite mixture model may incorporate a priori restrictions if they are available. An exclusion restriction in which one or more regressors affect one marginal mean but not the other is an example.

In the illustrative example of an FMM two-component Poisson model that follows, `educyr` (years of education) affects one component but not the other. Additionally, the mixing proportion is parameterized as a function of `actlim` and `totchr`, as before. The imposed restriction can be tested using the LR test.

```

. * FMM two-component Poisson: Different regressors in the two components
. fmm, lcprob(actlim totchr) nolog vce(robust):
>     (poisson docvis private age age2 educyr)
>     (poisson docvis private age age2)

Finite mixture model                                         Number of obs = 3,677
Log pseudolikelihood = -12105.773

```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1. Class	(base outcome)					
2. Class						
actlim	.3620486	.1010836	3.58	0.000	.1639284	.5601688
totchr	.6582347	.0384961	17.10	0.000	.5827838	.7336857
_cons	-2.276095	.1145746	-19.87	0.000	-2.500657	-2.051533

Class: 1
 Response: docvis
 Model: poisson

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
docvis						
private	.2109806	.0608503	3.47	0.001	.0917161	.3302451
age	.3539595	.1041269	3.40	0.001	.1498745	.5580445
age2	-.0022691	.0006901	-3.29	0.001	-.0036218	-.0009165
educyr	.0179365	.0055702	3.22	0.001	.007019	.0288539
_cons	-12.79023	3.909087	-3.27	0.001	-20.4519	-5.128563

Class: 2
 Response: docvis
 Model: poisson

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
docvis						
private	.1151981	.065195	1.77	0.077	-.0125817	.242978
age	.1919411	.1190989	1.61	0.107	-.0414884	.4253706
age2	-.0012686	.0007857	-1.61	0.106	-.0028086	.0002713
_cons	-4.583763	4.498159	-1.02	0.308	-13.39999	4.232467

We do not discuss the results because the example just illustrates the method of imposing restrictions.

20.5.10 Model selection

Choosing the best model involves tradeoffs between fit, parsimony, and ease of interpretation. Which of the six models that have been estimated best fits the data?

The LR test can be used to test the Poisson against the NB2 model for the basic model, the hurdle model, and the two-component FMM models. In each case, this leads to strong rejection of the Poisson model.

The remaining models are nonnested, so LR tests are not appropriate. Instead, we use various information criteria. Table 20.1 summarizes three commonly used model-comparison statistics—log likelihood and Akaike and Bayes information criteria (AIC and BIC)—explained in section 13.8.2. The log likelihood for the hurdle model is simply the sum of log likelihoods for the two parts of the model, whereas for the other models, it is directly given as command output.

Table 20.1. Goodness-of-fit criteria for six models

Model	Parameters	Log likelihood	AIC	BIC
Poisson	8	-15,019.64	30,055	30,105
NB2	9	-10,589.34	21,197	21,253
Poisson hurdle	16	-14,037.91	28,108	28,207
NB2 hurdle	17	-10,493.23	21,020	21,126
FMM two-component Poisson	17	-12,100.19	24,234	24,340
FMM two-component NB2	19	-10,534.26	21,106	21,224

All three criteria suggest that the NB2 hurdle model provides the best fitting and the most parsimonious specification. Such an unambiguous outcome is not always realized.

20.5.11 A cautionary note

It is easy to overparameterize mixture models. When the number of components is small, say, 2, and the means of the component distribution are far apart, clear discrimination between the components will emerge. However, if this is not the case, and a larger value of m is specified, unambiguous identification of all components may be difficult because of the increasing overlap in the distributions.

In particular, the presence of outliers may give rise to components that account for a small proportion of the observations. For example, if $m = 3$, $\pi_1 = 0.6$, $\pi_2 = 0.38$, and $\pi_3 = (1 - 0.6 - 0.38) = 0.02$, then this means that the third component accounts for just 2% of the data. If 2% of the sample is a small number, one might regard the result as indicating the presence of extreme observations.

There are a number of indications of failure of identification or fragile identification of mixture components. We list several examples. First, the log likelihood may increase only slightly when additional components are added. Second, the log likelihood may “fall” when additional components are added, which could be indicative of a multimodal objective function. Third, one or more mixture components may be small in the sense of accounting for few observations.

Finally, slow convergence is to be expected when the expectation maximization algorithm is used and identification of individual components is weak as suggested by a flat log likelihood. Therefore, it is advisable to use contextual knowledge and information when specifying, evaluating, and interpreting FMM output.

20.6 Zero-inflated models

The hurdle model treats zero counts as coming from a model that differs from that for positive counts. The motivation is that a hurdle needs to be crossed before the event of interest occurs. When the event occurs, it is modeled by a zero-truncated model for one or more event occurrences.

The related zero-inflated model also has a separate model for zeros but additionally allows that zeros come from a regular count model. The motivation is that a recording mechanism may occasionally fail to record the count, with zero substituting for the missing observation, so the regular count model is supplemented by a model for these additional zero counts.

The zero-inflated count model is a special case of a two-component FMM, a mixture of a binary outcome model and a count model. This special case is called a point-mass distribution, here with point mass at zero.

20.6.1 Data summary

The dataset used in this section overlaps heavily with that used in preceding sections of this chapter. It covers the same 3,677 individuals from a cross-sectional sample of persons aged 65 and higher from the U.S. Medical Expenditure Panel Survey for 2003.

The main change is that the dependent count variable is now the number of emergency room visits (`er`) by the survey respondent because this has many more zeros. An emergency room visit is a rare event for the Medicare elderly population who have access to care through their public insurance program and hence do not need to use emergency room facilities as the only available means of getting care.

The full set of explanatory variables in the model was initially the same as that used in the `docvis` example. However, after some preliminary analysis, this list was reduced to just three health-status variables—`age`, presence of activity limitation (`actlim`), and number of chronic conditions (`totchr`). The summary statistics follow, along with a tabulation of the frequency distribution for `er`.

```

. * Summary statistics for emergency room visits
. qui use mus220mepsemergroom, clear
. global xlist1 age i.actlim totchr

. summarize er $xlist1

```

Variable	Obs	Mean	Std. dev.	Min	Max
er	3,677	.2774001	.6929326	0	10
age	3,677	74.24476	6.376638	65	90
actlim					
0	3,677	.666848	.4714045	0	1
1	3,677	.333152	.4714045	0	1
totchr	3,677	1.843351	1.350026	0	8
. tabulate er					
# emergency room visits	Freq.	Percent	Cum.		
0	2,967	80.69	80.69		
1	515	14.01	94.70		
2	128	3.48	98.18		
3	40	1.09	99.27		
4	15	0.41	99.67		
5	8	0.22	99.89		
6	2	0.05	99.95		
7	1	0.03	99.97		
10	1	0.03	100.00		
Total	3,677	100.00			

Compared with `docvis`, the `er` variable has a much higher proportion (80.7%) of zeros. The first four values (0, 1, 2, 3) account for over 99% of the probability mass of `er`.

In itself, this does not imply that we have the “excess zero” problem. Given the mean value of 0.2774, a Poisson distribution predicts that $\Pr(Y = 0) = e^{-0.2774} = 0.758$. The observed proportion of 0.807 is higher than this, but the difference could potentially be explained by the regressors in the model. So there is no need to jump to the conclusion that a zero-inflated variant is essential.

20.6.2 Models for zero-inflated data

The zero-inflated model was originally proposed to handle data with excess zeros relative to the Poisson model. Like the hurdle model, it supplements a count density, $f_2(\cdot)$, with a binary process with a density of $f_1(\cdot)$. If the binary process takes on a value of 0, with a probability of $f_1(0)$, then $y = 0$. If the binary process takes on a value of 1, with a probability of $f_1(1)$, then y takes on the count values $0, 1, 2, \dots$ from the count density $f_2(\cdot)$. This lets zero counts occur in two ways: as a realization of the binary process and as a realization of the count process when the binary random variable takes on a value of 1.

Suppressing regressors for notational simplicity, the zero-inflated model has a density of

$$f(y) = \begin{cases} f_1(0) + \{1 - f_1(0)\}f_2(0) & \text{if } y = 0 \\ \{1 - f_1(0)\}f_2(y) & \text{if } y \geq 1 \end{cases}$$

As in the case of the hurdle model, the probability $f_1(0)$ may be a constant or may be parameterized through a binomial model like the logit or probit. Once again, the set of variables in the $f_1(\cdot)$ density need not be the same as those in the $f_2(\cdot)$ density.

To estimate the parameters of the zero-inflated Poisson (ZIP) and zero-inflated NB (ZINB) models, we use the estimation commands `zip` and `zinb`, respectively. The partial syntax for `zip` is

```
zip depvar [indepvars] [if] [in] [weight],
inflate(varlist[, offset(varname)] | _cons) [options]
```

where `inflate(varlist)` specifies the variables, if any, that determine the probability that the count is logit (the default) or probit (the `probit` option). Other options are essentially the same as for `poisson`.

The partial syntax for `zinb` is essentially the same as that for `zip`. Other options are the same as for `nbreg`. The only NB model fit is an NB2 model.

For the Poisson and NB models, the count process has conditional mean $\exp(\mathbf{x}_2' \boldsymbol{\beta}_2)$, and the corresponding with-zeros model can be shown to have

conditional mean

$$E(y|\mathbf{x}) = \{1 - f_1(0|\mathbf{x}_1)\} \times \exp(\mathbf{x}'_2 \boldsymbol{\beta}_2) \quad (20.10)$$

where $1 - f_1(0|\mathbf{x}_1)$ is the probability that the binary process variable equals 1. The MES are complicated by the presence of regressors in both parts of the model, as for the hurdle model. But if the binary process does not depend on regressors, so $f_1(0|\mathbf{x}_1) = f_1(0)$, then the parameters, $\boldsymbol{\beta}_2$, can be directly interpreted as semielasticities, as for the regular Poisson and NB models.

After the `zip` and `zinb` commands, the predicted mean in (20.10) can be obtained by using the `predict` postestimation command, and the `margins` command can be used to obtain the AME, MEM, or MER.

20.6.3 Results for the NB2 model

Before fitting a zero-inflated model, we apply the NB2 model to emergency room visits.

```
. * NB2 for emergency room visits
. nbreg er $xlist1, nolog vce(robust)

Negative binomial regression
Number of obs = 3,677
Wald chi2(3) = 210.17
Prob > chi2 = 0.0000
Pseudo R2 = 0.0464

Dispersion: mean
Log pseudolikelihood = -2314.4927
```

er	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
1.actlim	.0088528	.006115	1.45	0.148	-.0031324	.0208381
	.6859572	.0865077	7.93	0.000	.5164052	.8555091
	.2514885	.0297963	8.44	0.000	.1930889	.3098882
	-2.799848	.4636337	-6.04	0.000	-3.708554	-1.891143
/lnalpha	.4464685	.1299578			.1917558	.7011812
alpha	1.562783	.2030959			1.211375	2.016133

There is statistically significant overdispersion with $\alpha = 1.56$. The coefficient estimates are similar to those for the Poisson model (not given).

The regression equation has low but statistically significant explanatory power. For an event that is expected to have a high degree of inherent randomness, low overall explanatory power is to be expected. Having an activity limitation and a high number of chronic conditions is positively associated with `er` visits.

20.6.4 Results for the ZINB model

The parameters of the ZINB model are estimated by using the `zinb` command. We use the same set of regressors in the two parts of the model.

```
. * Zero-inflated NB2 for emergency room visits
. zinb er $xlist1, inflate($xlist1) nolog vce(robust)

Zero-inflated negative binomial regression
Inflation model: logit
Number of obs = 3,677
Nonzero obs = 710
Zero obs = 2,967
Wald chi2(3) = 22.81
Prob > chi2 = 0.0000
Log pseudolikelihood = -2304.868
```

er	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
er						
age	.0035485	.0075037	0.47	0.636	-.0111585	.0182555
1.actlim	.2743106	.2278723	1.20	0.229	-.1723109	.7209321
totchr	.1963408	.0727287	2.70	0.007	.0537951	.3388866
_cons	-1.822978	.6681715	-2.73	0.006	-3.13257	-.5133861
inflate						
age	-.0236763	.0281379	-0.84	0.400	-.0788255	.0314729
1.actlim	-4.22705	22.47385	-0.19	0.851	-48.27499	39.82089
totchr	-.3471091	.2899582	-1.20	0.231	-.9154167	.2211986
_cons	1.846526	2.039352	0.91	0.365	-2.150531	5.843584
/lnalpha	.1602371	.231154	0.69	0.488	-.2928164	.6132905
alpha	1.173789	.271326			.7461591	1.846497

The estimated coefficients differ from those from the NB2 model. The two models have different conditional means—see ([20.10](#))—so the coefficients are not directly comparable.

The first set of output for the NB2 component implies that `er` increases as the regressors increase. The second set of output for the logit model

component shows that the probability of a zero decreases as the regressors increase, which again implies an increase in `er` as the regressors increase. More generally, there need not be such consistency of coefficient signs across the two components.

ME for ZINB model

The ZINB model estimates are most easily interpreted by obtaining the AMES using the `margins` command. We obtain

```
. * ZINB: AMEs
. qui zinb er $xlist1, inflate($xlist1) vce(robust)
. margins, dydx(*)

Average marginal effects                                         Number of obs = 3,677
Model VCE: Robust
Expression: Predicted number of events, predict()
dy/dx wrt: age 1.actlim totchr
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
age	.0020337	.0016783	1.21	0.226	-.0012557	.005323
	.2032745	.0267769	7.59	0.000	.1507928	.2557563
	.0698286	.0086039	8.12	0.000	.0529651	.086692
1.actlim						
totchr						

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

By comparison, the AMES for the NB2 models are

```
. * NB2: AMEs
. qui nbreg er $xlist1, vce(robust)
. margins, dydx(*)

Average marginal effects                                         Number of obs = 3,677
Model VCE: Robust
Expression: Predicted number of events, predict()
dy/dx wrt: age 1.actlim totchr
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
age	.0024632	.0017054	1.44	0.149	-.0008793	.0058057
	.1961981	.0259474	7.56	0.000	.1453421	.2470541
	.0699732	.0088665	7.89	0.000	.0525951	.0873513
1.actlim						
totchr						

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

The AMES for the statistically insignificant regressor `age` differ across the two models, while the AMES for `actlim` and `totchr` are very similar.

20.6.5 Point-mass version of zip and zinb commands

Exploiting the insight that a point-mass distribution is also a finite mixture model, Stata's `fmm` prefix now provides an alternative to `zip` and `zinb` commands.

```
. * Zero-inflated NB2 estimated using fmm command with point-mass zero
. fmm, nolog vce(robust): (pointmass er, lcprob($xlist1)) (nbreg er $xlist1)
Finite mixture model                                         Number of obs = 3,677
Log pseudolikelihood = -2304.8677
```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
1.Class					
age	-.023672	.028149	-0.84	0.400	-.0788431 .0314991
1.actlim	-4.219059	23.91722	-0.18	0.860	-51.09596 42.65784
totchr	-.3470913	.2964756	-1.17	0.242	-.9281729 .2339903
_cons	1.84657	2.041192	0.90	0.366	-2.154094 5.847233
2.Class	(base outcome)				

```
Class: 2
Response: er
Model: nbreg, dispersion(mean)
```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
er					
age	.003547	.0075502	0.47	0.639	-.0112511 .018345
1.actlim	.2742822	.2332858	1.18	0.240	-.1829496 .731514
totchr	.1963124	.0760157	2.58	0.010	.0473244 .3453003
_cons	-1.822645	.6810172	-2.68	0.007	-3.157414 -.4878756
/er					
lnalpha	.1599948	.2435035			-.3172633 .6372529

The results are the same as those from the `zinb` command, aside from a computational difference that disappears by setting the `ltolerance()` option to a smaller value than program defaults of `1e-7`.

The postestimation commands `estat lcmean` and `estat lcprob` provide summary statistics.

```
. * Zero-inflated NB2 estimated using fmm command: Summary statistics
. predict mu*
(option mu assumed)
. summarize mu*
```

Variable	Obs	Mean	Std. dev.	Min	Max
mu1	3,677	.3516795	.1333575	.2034994	1.315176

```
. estat lcmean
Latent class marginal means                                         Number of obs = 3,677
Expression: Predicted mean (# emergency room visits in class 2.Class),
predict(outcome(er) class(2))
```

	Delta-method				
	Margin	std. err.	z	P> z	[95% conf. interval]
2 er	.3516795	.0468653	7.50	0.000	.2598253 .4435337

```
. estat lcprob
Latent class marginal probabilities                               Number of obs = 3,677
```

	Delta-method		
	Margin	std. err.	[95% conf. interval]
Class 1	.2722706	.1017516	.1202945 .5058444
2	.7277294	.1017516	.4941556 .8797055

The excess zeros arise with average probability 0.272, the regular NB2 count process arises with average probability 0.728, and the regular process counts have average 0.352. So the average prediction is

$0.272 \times 0 + 0.728 \times 0.352 = 0.256$, compared with the sample average 0.277.

20.6.6 Model comparison

Testing the ZINB model against the simpler NB model, is a nonstandard test. Versions up to Stata 14 provided an option to implement the LR test of

[Vuong \(1989\)](#) to discriminate between the NB and ZINB models. However, this test is not appropriate because the ZINB model reduces to the NB model only at the boundary of the parameter space for the logit model [so that $f_1(0) = 0$]; see [Wilson \(2015\)](#). Similar issues arise for testing the ZIP model against the simpler Poisson model.

Instead, we compare the models on the basis of average predicted probabilities and information criteria.

In principle, model comparison is made simple by applying the community-contributed `countfit` command. Ideally, the command

```
* Comparison of NB and ZINB using countfit  
countfit er $xlist1, nbreg zinb nograph noestimates
```

fits both NB2 and ZINB models and gives average fitted frequencies and information criteria for the two models. At the time of writing, however, this command does not work for the `zinb` option, because it uses the Vuong test, which Stata no longer supports.

Instead, we use two separate and distinct commands. For the NB2 model, we use `countfit` for average fitted and actual frequencies, along with `estat ic`. We obtain

```
. * NB2 model: Average fitted frequencies and information criteria
. countfit er age actlim totchr, nbreg nograph noestimates nofit
Comparison of Mean Observed and Predicted Count
```

Model	Maximum Difference	At Value	Mean	Diff
NBRM	0.001	1	0.000	
NBRM: Predicted and actual probabilities				
Count	Actual	Predicted	Diff	Pearson
0	0.807	0.807	0.000	0.001
1	0.140	0.139	0.001	0.047
2	0.035	0.036	0.001	0.053
3	0.011	0.011	0.000	0.040
4	0.004	0.004	0.000	0.001
5	0.002	0.002	0.001	0.558
6	0.001	0.001	0.000	0.181
7	0.000	0.000	0.000	0.052
8	0.000	0.000	0.000	0.610
9	0.000	0.000	0.000	0.308
Sum	1.000	1.000	0.004	1.850

```
. qui nbreg er $xlist1
```

```
. estat ic
```

Akaike's information criterion and Bayesian information criterion

Model	N	ll(null)	ll(model)	df	AIC	BIC
.	3,677	-2427.068	-2314.493	5	4638.985	4670.035

Note: BIC uses N = number of observations. See [R] BIC note.

The average predicted probabilities are close to actual frequencies.

For the ZINB model, we use the `zinb` command, followed by the `prcounts` command for average fitted probabilities, along with `estat ic`. We obtain

```

. * ZINB model: Average fitted frequencies and information criteria
. qui zinb er $xlist1, inflate($xlist1) vce(robust)
. prcounts erpr, max(9)
. summarize erprpr*

```

Variable	Obs	Mean	Std. dev.	Min	Max
erprpr0	3,677	.8083271	.0967287	.451979	.9293385
erprpr1	3,677	.1345371	.0549748	.0579817	.2367817
erprpr2	3,677	.038742	.0246506	.0103499	.1310115
erprpr3	3,677	.0120868	.0108068	.0018967	.0755851
erprpr4	3,677	.0040225	.0047706	.0003521	.0441736
erprpr5	3,677	.001416	.0021523	.0000659	.0260145
erprpr6	3,677	.0005238	.000999	.0000124	.0153982
erprpr7	3,677	.0002026	.0004783	2.34e-06	.0091473
erprpr8	3,677	.0000815	.0002362	4.42e-07	.0054486
erprpr9	3,677	.000034	.0001201	8.38e-08	.0032523
erprprgt	3,677	.0000267	.0001371	-1.19e-07	.0048559

```
. estat ic
```

Akaike's information criterion and Bayesian information criterion

Model	N	ll(null)	ll(model)	df	AIC	BIC
.	3,677	-2322.013	-2304.868	9	4627.735	4683.624

Note: BIC uses N = number of observations. See [R] BIC note.

The average predicted probabilities from the ZINB model are not as close to the actual frequencies as those from the NB2 model.

The AIC statistic favors the ZINB model, whereas BIC, which has a larger penalty for model size, favors the NB2 model (as $4670 < 4683$).

This example indicates that having many zeros in the dataset does not automatically mean that a zero-inflated model is necessary. For these data, the ZINB model is only a slight improvement on the NB2 model and is actually no improvement at all if BIC is used as the model-selection criterion. Furthermore, the NB2 model has the advantage of easier interpretation of parameter estimates.

20.7 Endogenous regressors

So far, the regressors in the count regression are assumed to be exogenous. We now consider a more general model in which a regressor is endogenous. Specifically, the empirical example used in this chapter has assumed that the regressor `private` is exogenous. But individuals can and do choose whether they want supplementary private insurance, and hence potentially this variable is endogenous, that is, jointly determined with `docvis`. If endogeneity is ignored, the standard single-equation estimator will be inconsistent.

The general issues are similar to those already presented in section [17.9](#) for endogeneity in the probit model. [Cameron and Trivedi \(2005\)](#), chap. 20.6) and especially [Wooldridge \(2010\)](#), chap. 18.5) provide details.

There are two distinct general approaches. One approach specifies a structural equation for the count outcome (y_1) (usually with an exponential conditional mean function) and first-stage equations for endogenous regressors, say, y_2 . The model is assumed to be recursive in the sense that there is no direct feedback from y_2 to y_1 ; see section 7.2.3. If the joint distribution of y_1 and y_2 can be parametrically specified, then estimation is by ML. Alternatively, a control function approach enables two-step estimation under weaker assumptions.

An alternative, less parametric approach is nonlinear instrumental variables (IV) or generalized method of moments (GMM) based on orthogonality of instrumental variables with the “error” term in the structural model. This approach is applicable to any dependent variable for which an exponential conditional mean model is appropriate and for endogenous regressors that need not be continuous.

20.7.1 ML estimator of structural model

The structural-model approach defines explicit models for both the dependent variable of interest (y_1) and the endogenous regressor (y_2).

Model and assumptions

The structural equation for the count outcome is a Poisson model for y_1 with a mean that depends on an endogenous regressor y_2 . Specifically,

$$y_{1i} \sim \text{Poisson}(\mu_i)$$

$$\mu_i = E(y_{1i}|y_{2i}, \mathbf{x}_{1i}, u_{1i}) = \exp(\beta_1 y_{2i} + \mathbf{x}'_{1i} \boldsymbol{\beta}_2 + u_{1i}) \quad (20.11)$$

This model introduces an error term u_1 that is uncorrelated with a vector of exogenous variables \mathbf{x}_1 but is potentially correlated with y_2 , in which case y_2 is endogenous and the usual Poisson quasi-ML estimator is inconsistent.

A linear first-stage equation is specified for y_2 , with

$$y_{2i} = \mathbf{x}'_{1i} \boldsymbol{\gamma}_1 + \mathbf{x}'_{2i} \boldsymbol{\gamma}_2 + \varepsilon_i \quad (20.12)$$

where \mathbf{x}_2 is a vector of exogenous variables that affects y_2 nontrivially but does not directly affect y_1 and hence is an independent source of variation in y_2 . It is standard to refer to this as an exclusion restriction and to refer to \mathbf{x}_2 as excluded exogenous variables or instrumental variables. By convention, a condition for robust identification of (20.11), as in the case of the linear model, is that there is available at least one valid excluded variable (instrument). When only one such variable is present in (20.12), the model is said to be just identified, and it is said to be overidentified if there are additional excluded variables.

Endogeneity arises because the errors u_1 and ε are correlated. Specifically, assume that

$$u_{1i} = \rho \varepsilon_i + \eta_i \quad (20.13)$$

where $\eta_i \sim [0, \sigma_\eta^2]$ is independent of $\varepsilon_i \sim [0, \sigma_\varepsilon^2]$. Then ε is a common latent factor that affects both y_1 and y_2 and is the only source of dependence between them, after controlling for the influence of the observable variables \mathbf{x}_1 and \mathbf{x}_2 . The variable y_2 is endogenous in the model for y_1 , unless $\rho = 0$, because both y_2 and u_1 appear in the model (20.11) and both depend on ε , via (20.12) and (20.13).

The error term u_1 can be interpreted as unobserved heterogeneity that has a multiplicative effect on the conditional mean because $\mu_i = \exp(\beta_1 y_{2i} + \mathbf{x}'_{1i} \boldsymbol{\beta}_2) \times e^{u_{1i}}$. Note that the error term not only induces endogeneity in (20.13) but also induces overdispersion, so that the Poisson model has been generalized to control for overdispersion as would be the case if an NB model was used.

ML estimation

ML estimation of this model, under the additional assumption of normally distributed errors ε_i and η_i , can be obtained using the `gsem` command detailed in section 23.6.

For the specific application outlined below, complete ML estimates are presented in section 23.6.5 and are not reproduced here. We merely note that ML estimation leads to the endogenous regressor `private` having coefficient 0.633 with standard error 0.219.

20.7.2 Control function estimator of structural equation

A control function estimator, or residual-augmentation estimator, obtains consistent estimates of the structural equation by adding an appropriate residual term as an additional regressor. For the current count application, it relies only on appropriate conditional mean assumptions and does not require the Poisson and normal distributional assumptions of the preceding ML approach.

Substituting (20.13) for u_1 in (20.11) yields $\mu = \exp(\beta_1 y_2 + \mathbf{x}'_1 \boldsymbol{\beta}_2 + \rho \varepsilon) e^\eta$. Taking the expectation with respect to η yields $E_\eta(\mu) = \exp(\beta_1 y_2 + \mathbf{x}'_1 \boldsymbol{\beta}_2 + \rho \varepsilon) \times E(e^\eta) = \exp(\beta_1 y_2 + \ln E(e^\eta) + \mathbf{x}'_1 \boldsymbol{\beta}_2 + \rho \varepsilon)$. The constant term $\ln E(e^\eta)$ can be absorbed in the coefficient of the intercept, a component of \mathbf{x}_1 . It follows that

$$\mu_i | \mathbf{x}_{1i}, y_{2i}, \varepsilon_i = \exp(\beta_1 y_{2i} + \mathbf{x}'_{1i} \boldsymbol{\beta}_2 + \rho \varepsilon_i) \quad (20.14)$$

where ε_i is a new additional variable.

If ε were observable, including it as a regressor would control for the endogeneity of y_2 . Given that it is unobservable, the estimation strategy is to replace it by a consistent estimate.

The following two-step estimation procedure is used: First, fit the first-stage model (20.12) by OLS, and generate the residuals $\widehat{\varepsilon}_i$. This requires the assumption that an instrument exists and that (20.12) is correctly specified so that $E(\varepsilon_i | \mathbf{x}_{1i}, \mathbf{x}_{2i}) = 0$. Second, estimate parameters of the Poisson model given in (20.14) after replacing ε_i by $\widehat{\varepsilon}_i$. As discussed below, if $\rho = 0$, then we can use the `vce(robust)` option, but if $\rho \neq 0$, then the VCE needs to be estimated with the bootstrap method detailed in section 12.4.5 that controls for the estimation of ε_i by $\widehat{\varepsilon}_i$ or by a method similar to that in section 13.3.11.

Application

We apply this two-step procedure to the Poisson model for the doctor-visits data analyzed in section 20.3, with the important change that `private` is now treated as endogenous.

Two excluded variables used as instruments are `income` and `ssiratio`. The first is a measure of total household income, and the second is the ratio of social security income to total income. Jointly, the two variables reflect the affordability of private insurance. A high value of `income` makes private insurance more accessible, whereas a high value of `ssiratio` indicates an income constraint and is expected to be negatively associated with `private`. For these to be valid instruments, we need to assume that for people aged 65–90 years, doctor visits are not determined by `income` or `ssiratio`, after controlling for other regressors that include a quadratic in age, education, health-status measures, and access to Medicaid.

The first step generates residuals from a linear probability regression of `private` on regressors and instruments.

```

. * Poisson control function estimator: First-stage linear regression
. qui use mus220mepsdocvis, clear
. global xlist2 medicaid age age2 educyr actlim totchr
. regress private $xlist2 income ssiratio, vce(robust)

Linear regression
Number of obs      =      3,677
F(8, 3668)         =     249.61
Prob > F          =     0.0000
R-squared           =     0.2108
Root MSE            =     .44472

```

private	Coefficient	Robust				
		std. err.	t	P> t	[95% conf. interval]	
medicaid	-.3934477	.0173623	-22.66	0.000	-.4274884	-.3594071
age	-.0831201	.0293734	-2.83	0.005	-.1407098	-.0255303
age2	.0005257	.0001959	2.68	0.007	.0001417	.0009098
educyr	.0212523	.0020492	10.37	0.000	.0172345	.02527
actlim	-.0300936	.0176874	-1.70	0.089	-.0647718	.0045845
totchr	.0185063	.005743	3.22	0.001	.0072465	.0297662
income	.0027416	.0004736	5.79	0.000	.0018131	.0036702
ssiratio	-.0647637	.0211178	-3.07	0.002	-.1061675	-.0233599
_cons	3.531058	1.09581	3.22	0.001	1.3826	5.679516

```
. predict lpuhat, residual
```

The two instruments, `income` and `ssiratio`, are highly statistically significant with expected signs.

The second step fits a Poisson model on regressors that include the first-step residual.

```

. * Poisson control function estimator: Second-stage poisson regression
. poisson docvis private $xlist2 lpuhat, vce(robust) nolog

Poisson regression
Number of obs = 3,677
Wald chi2(8) = 718.87
Prob > chi2 = 0.0000
Pseudo R2 = 0.1303

Log pseudolikelihood = -15010.614

```

docvis	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
private	.5505541	.2453175	2.24	0.025	.0697407	1.031368
medicaid	.2628822	.1197162	2.20	0.028	.0282428	.4975217
age	.3350604	.0696064	4.81	0.000	.1986344	.4714865
age2	-.0021923	.0004576	-4.79	0.000	-.0030893	-.0012954
educyr	.018606	.0080461	2.31	0.021	.002836	.034376
actlim	.2053417	.0414248	4.96	0.000	.1241505	.286533
totchr	.24147	.0129175	18.69	0.000	.2161523	.2667878
lpuhat	-.4166838	.249347	-1.67	0.095	-.9053949	.0720272
_cons	-11.90647	2.661445	-4.47	0.000	-17.1228	-6.69013

The z statistic for the coefficient of `lpuhat` provides the basis for a robust Wald test of the null hypothesis of exogeneity, $H_0: \rho = 0$. The z statistic has a p -value of 0.095 against $H_1: \rho \neq 0$, leading to nonrejection of H_0 at the 0.05 level. But a one-sided test against $H_1: \rho < 0$ may be appropriate because this was proposed on a priori grounds. Then the p -value is 0.048, leading to rejection of H_0 at the 0.05 level.

Bootstrapped standard errors

If $\rho \neq 0$, then the VCE of the second-step estimator needs to be adjusted for the replacement of ε_i with $\widehat{\varepsilon}_i$. One way to do so is to adapt the method given in section 13.3.11 for the linear model. Here we instead bootstrap as in section 12.4.5. We have

```

. * Poisson control function estimator: Obtain bootstrap standard errors
> for estimator
. program endogtwostep, eclass
1.      version 17
2.      tempfile b
3.      capture drop lpuhat2
4.      regress private $xlist2 income ssiratio
5.      predict lpuhat2, residual
6.      poisson docvis private $xlist2 lpuhat2
7.      matrix `b' = e(b)
8.      ereturn post `b'
9. end
. bootstrap _b, reps(400) seed(10101) nodots nowarn: endogtwostep
Bootstrap results                                         Number of obs = 3,677
                                                               Replications = 400

```

	Observed coefficient	Bootstrap std. err.	z	P> z	Normal-based [95% conf. interval]	
private	.5505541	.2799587	1.97	0.049	.0018452	1.099263
medicaid	.2628822	.1284541	2.05	0.041	.0111169	.5146476
age	.3350604	.0744832	4.50	0.000	.189076	.4810449
age2	-.0021923	.0004888	-4.48	0.000	-.0031504	-.0012342
educyr	.018606	.0091275	2.04	0.042	.0007164	.0364957
actlim	.2053417	.0435134	4.72	0.000	.1200571	.2906264
totchr	.24147	.0134558	17.95	0.000	.2150971	.267843
lpuhat2	-.4166838	.2827737	-1.47	0.141	-.9709101	.1375424
_cons	-11.90647	2.851588	-4.18	0.000	-17.49548	-6.317456

The standard errors differ little from the previous standard errors obtained by using the option `vce(robust)`. From section [20.3.2](#), the Poisson ML estimate of the coefficient on `private` was 0.142 with a robust standard error of 0.036. The two-step estimate of the coefficient on `private` is 0.551 with a standard error of 0.245. (The ML estimate given in section [23.6.5](#) is 0.633 with standard error 0.219.)

The precision of estimation is much less than the exogenous case, with a standard error that is seven times larger. This large increase is very common for cross-sectional data, where instruments are not very highly correlated with the regressor being instrumented. At the same time, the coefficient is four times larger, and so the regressor retains statistical significance. The effect is now very large, with private insurance leading to a $100(e^{0.551} - 1) = 73\%$ increase in doctor visits.

The negative coefficient of lpuhat2 can be interpreted to mean that the latent factor, which increases the probability of purchasing private insurance lowers the number of doctor visits—an effect consistent with favorable selection, according to which the relatively healthy individuals self-select into insurance. Controlling for endogeneity has a substantial effect on the ME of an exogenous change in private insurance because the coefficient of private and the associated MES are now much higher.

20.7.3 Nonlinear IV (GMM) estimators

An alternative method for controlling for endogeneity is nonlinear IV or GMM based on a moment condition that does not require specification of model or error distributions.

The literature on count-data models makes a case for both additive and multiplicative forms of heterogeneity; in turn, these imply different moment functions and hence potentially different GMM estimates. See, for example, [Cameron and Trivedi \(2005\)](#), chap. 20.6 for explanation.

The nonlinear IV presented in section [16.8](#) can be motivated as follows. Suppose $y_{1i} = \exp(\beta_1 y_{2i} + \mathbf{x}'_{1i} \boldsymbol{\beta}_2) + u_{1i}$, so the error term u_{1i} has an additive effect. Then $u_{1i} = y_i - \exp(\beta_1 y_{2i} + \mathbf{x}'_{1i} \boldsymbol{\beta}_2)$, and we assume existence of instruments \mathbf{z}_i satisfying $E(u_{1i} | \mathbf{z}_i) = 0$ and hence $E(\mathbf{z}_i u_{1i}) = \mathbf{0}$. This leads to moment condition

$$E[\mathbf{z}_i \{y_{1i} - \exp(\beta_1 y_{2i} + \mathbf{x}'_{1i} \boldsymbol{\beta}_2)\}] = \mathbf{0} \quad (20.15)$$

where the instruments $\mathbf{z}_i = (\mathbf{x}'_{1i}, \mathbf{x}'_{2i})'$, with \mathbf{x}_{2i} being additional variables necessary for identification.

[Mullahy \(1997\)](#) noted that for count data and, more generally, for an exponential conditional mean, it is more natural to work with a multiplicative error, as in [\(20.11\)](#). Let $y_{1i} = \exp(\beta_1 y_{2i} + \mathbf{x}'_{1i} \boldsymbol{\beta}_2)w_{1i}$. Assume there are instruments \mathbf{z}_i satisfying $E(w_{1i} | \mathbf{z}_i) = 1$, where there is no loss in generality in setting the expectation to 1 if \mathbf{x}_{1i} includes a constant. Now $E(w_{1i} - 1 | \mathbf{z}_i) = 1$ by assumption and $w_{1i} = y_{1i} / \exp(\beta_1 y_{2i} + \mathbf{x}'_{1i} \boldsymbol{\beta}_2)$

, so $E\{y_{1i}/\exp(\beta_1 y_{2i} + \mathbf{x}'_{1i}\boldsymbol{\beta}_2) - 1|\mathbf{z}_i\} = 1$. This leads to moment condition

$$E[\mathbf{z}_i \{y_{1i}/\exp(\beta_1 y_{2i} + \mathbf{x}'_{1i}\boldsymbol{\beta}_2) - 1\}] = \mathbf{0} \quad (20.16)$$

where again the instruments $\mathbf{z}_i = (\mathbf{x}'_{1i}, \mathbf{x}'_{2i})'$, with \mathbf{x}_{2i} being additional variables necessary for identification.

Equations (20.11)–(20.13) do not imply either (20.15) or (20.16), so this less parametric approach will lead to an estimator that differs from that using the structural approach even in the limit as $N \rightarrow \infty$. This approach seems more appropriate especially when y_{2i} is binary and therefore less likely to satisfy assumption (20.12).

ivpoisson gmm command

These nonlinear IV estimators can be implemented using the `ivpoisson gmm` command, which has basic syntax similar to the `ivregress` command, and the weight matrix options of the `gmm` command. The default is an additive error, while the `multiplicative` option is used for a multiplicative error. The default is for two-step estimation, while the `onestep` option is used for one-step GMM estimation. Two-step estimation is relevant for an overidentified model, and in that case the `estat overid` postestimation command performs an overidentifying restrictions test. The default is for `ivpoisson` to report heteroskedastic–robust standard errors.

The following Stata commands compute one-step and two-step estimators for the additive and multiplicative error models. For all estimators, we use the same instruments as before.

```

. * ivpoisson gmm: (1)-(2) additive one and two step; (3)-(4) multiplicative
> one and two step
. qui ivpoisson gmm docvis $xlist2 (private=income ssiratio), onestep
. estimates store GMMadd1S
. qui ivpoisson gmm docvis $xlist2 (private=income ssiratio)
. estimates store GMMadd2S
. qui ivpoisson gmm docvis $xlist2 (private=income ssiratio), onestep multiplicative
. estimates store GMMmult1S
. qui ivpoisson gmm docvis $xlist2 (private=income ssiratio), multiplicative
. estimates store GMMmult2S

```

The estimates are as follows:

```

. * Table for GMM additive one step and two step and multiplicative
> one step and two step
. estimates table GMMadd1S GMMadd2S GMMmult1S GMMmult2S,
> b(%7.4f) se(%7.3f) stats(N) stfmt(%9.1f) modelwidth(9)

```

Variable	GMMadd1S	GMMadd2S	GMMmult1S	GMMmult2S
private	0.5921	0.5864	0.7912	0.7876
	0.340	0.341	0.280	0.279
medicaid	0.3187	0.2876	0.3290	0.3010
	0.191	0.191	0.110	0.109
age	0.3323	0.3496	0.3534	0.3698
	0.071	0.071	0.076	0.075
age2	-0.0022	-0.0023	-0.0023	-0.0024
	0.000	0.000	0.001	0.000
educyr	0.0191	0.0174	0.0122	0.0113
	0.009	0.009	0.009	0.008
actlim	0.2085	0.1843	0.2042	0.1908
	0.043	0.042	0.043	0.042
totchr	0.2418	0.2461	0.2855	0.2893
	0.013	0.013	0.015	0.015
_cons	-11.8635	-12.5396	-12.7527	-13.3824
	2.733	2.738	2.876	2.836
N	3677	3677	3677	3677

Legend: b/se

The results are qualitatively similar to preceding results. The one-step and two-step estimates are very similar with similar precision. The coefficient of the endogenous regressor `private` is larger, and is more precisely estimated, with a multiplicative error.

The overidentifying restrictions test presented for the linear IV model in section 7.4.8 extends to nonlinear GMM models. For this example, following two-step GMM estimation of the multiplicative error model, we have

```
. * Test of overidentifying restriction following two-step GMM
. qui ivpoisson gmm docvis $XLISTEX0G (private=income ssiratio), multiplicative
. estat overid
Test of overidentifying restriction:
Hansen's J chi2(1) = .390236 (p = 0.5322)
```

The test statistic is $\chi^2(1)$ distributed because there is one overidentifying restriction (`income` and `ssiratio` are instruments for `private`, and all other regressors are instruments for themselves). Because $p > 0.05$, we do not reject the null hypothesis and conclude that the overidentifying restriction is valid.

The same models could be fit using the `gmm` command. For one-step estimation, for example, the commands are

```
. * gmm command for additive and multiplicative one step
. gmm (docvis - exp({xb:private $xlist2 _cons})),
>     instruments(income ssiratio $xlist2) onestep vce(robust)
  (output omitted)
. gmm ( (docvis / exp({xb:private $xlist2 _cons})) - 1),
>     instruments(income ssiratio $xlist2) onestep vce(robust)
  (output omitted)
```

The estimates and standard errors, not reported here, are identical to those obtained using the `ivpoisson gmm` command.

ivpoisson efunction command

A variant of the preceding GMM approach with a multiplicative error introduces a relationship between the structural equation and first-stage model errors, leading to the addition of the predicted residual from the first-stage regression as an additional variable in moment condition (20.16).

Assume that the errors in (20.11) and (20.12) are related by $u_{1i} = \rho\varepsilon_i + \eta_i$, as in (20.13), where η_i is independent. Then the moment condition becomes

$$E [\mathbf{z}_i \{y_{1i} / \exp(\beta_1 y_{2i} + \mathbf{x}'_{1i} \boldsymbol{\beta}_2 + \rho \varepsilon_i) - 1\}] = \mathbf{0} \quad (20.17)$$

where ε_i is estimated by the residual from first-stage regression based on (20.12). This is implemented by replacing ε_i in (20.17) with $\varepsilon_i = y_{2i} - \mathbf{z}'_i \boldsymbol{\gamma}_1$, where $\mathbf{z}_i = (\mathbf{x}'_{1i}, \mathbf{x}'_{2i})'$, and adding a second moment condition $E\{\mathbf{z}_i(y_{2i} - \mathbf{z}'_i \boldsymbol{\gamma})\} = \mathbf{0}$ for the first-stage estimation.

`ivpoisson cfunction` provides one-step GMM estimates of this model. The term `cfunction` is used because it is a control function approach, but note that the control function approach is being applied to a different model than that in section 20.7.2. We obtain

```

. * ivpoisson cfunction estimation for endogenous Poisson
. ivpoisson cfunction docvis medicaid age age2 educyr actlim totchr
>           (private=income ssiratio), vce(robust)

Step 1
Iteration 0: GMM criterion Q(b) = .00243925
Iteration 1: GMM criterion Q(b) = 1.982e-06
Iteration 2: GMM criterion Q(b) = 1.137e-12
Iteration 3: GMM criterion Q(b) = 5.235e-25
note: model is exactly identified.

Exponential mean model with endogenous regressors

Number of parameters = 18                               Number of obs = 3,677
Number of moments = 18
Initial weight matrix: Unadjusted
GMM weight matrix: Robust

```

docvis	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
docvis						
medicaid	.3158814	.1232552	2.56	0.010	.0743057	.5574571
age	.3507938	.0724955	4.84	0.000	.2087052	.4928823
age2	-.0022849	.0004785	-4.78	0.000	-.0032227	-.0013471
educyr	.0145113	.0084008	1.73	0.084	-.0019538	.0309765
actlim	.215784	.0423114	5.10	0.000	.1328551	.2987128
totchr	.2770506	.0145983	18.98	0.000	.2484385	.3056627
private	.7078763	.2641085	2.68	0.007	.1902332	1.225519
_cons	-12.6812	2.756927	-4.60	0.000	-18.08468	-7.277722
private						
medicaid	-.3934477	.017341	-22.69	0.000	-.4274355	-.35946
age	-.0831201	.0293374	-2.83	0.005	-.1406203	-.0256198
age2	.0005257	.0001956	2.69	0.007	.0001423	.0009092
educyr	.0212523	.0020467	10.38	0.000	.0172408	.0252637
actlim	-.0300936	.0176658	-1.70	0.088	-.0647179	.0045306
totchr	.0185063	.005736	3.23	0.001	.007264	.0297487
income	.0027416	.000473	5.80	0.000	.0018145	.0036687
ssiratio	-.0647637	.0210919	-3.07	0.002	-.1061032	-.0234243
_cons	3.531058	1.094469	3.23	0.001	1.385939	5.676177
/c_private	-.5478738	.2655137	-2.06	0.039	-1.068271	-.0274764

Instrumented: private

Instruments: medicaid age age2 educyr actlim totchr income ssiratio

The coefficient of the endogenous regressor `private` is 0.708, closest to the multiplicative GMM estimates, and has standard error 0.264, which is the smallest among the various estimators considered in this section.

20.7.4 Weak instruments

If instruments are weakly correlated with the endogenous regressor, then the usual asymptotic theory may perform poorly. Then alternative methods may be used. Inference for weak instruments for the linear model is presented in section 7.7.

The discussion there includes methods based on minimum distance estimation due to [Magnusson \(2010\)](#) that can be applied to a wide range of linear and nonlinear structural models. Magnusson's article provides simulations and an application to a Poisson model with endogenous regressor and weak instruments using both his proposed methods for a structural model and weak instrument methods due to [Stock and Wright \(2000\)](#) for nonlinear GMM.

20.8 Clustered data

Section 13.9 presented in some detail various models and methods that can be applied when observations in the same cluster are correlated and observations in different clusters are uncorrelated. That discussion was illustrated using the Poisson model. Here we provide a brief summary that covers the broad range of count models, and not just the Poisson model.

The simplest approach is to continue to use the `poisson` and `nbreg` commands but to use the `vce(cluster clustvar)` option. This maintains the assumption that the conditional mean for individual i in cluster g is $E(y_{ig}|\mathbf{x}_{ig}) = \exp(\mathbf{x}'_{ig}\boldsymbol{\beta})$, the essential condition for consistency of these estimators. But it provides corrected standard errors that adjust for the loss in precision that arises because of observations no longer being independent within cluster. As with any cluster-robust estimate of standard errors, it is assumed that there are many clusters. Similarly, the nonlinear IV methods for models with endogenous regressors can continue to be used, along with the `vce(cluster clustvar)` option.

Potentially, more efficient estimates can be obtained by estimating using nonlinear feasible generalized least squares, assuming equicorrelation within cluster. For the Poisson model, this population-averaged approach uses the `xtgee` command with options `family(poisson)`, `link(log)`, and `corr(exchangeable)`. It is good practice to add the `vce(robust)` option because this provides cluster-robust standard errors that guard against within-cluster correlation not being exactly one of equicorrelation. An equivalent command is `xtpoisson, pa corr(exchangeable) vce(robust)`. One must first use the `xtset` command to define the cluster variable. Similarly, one can use the command `xtnbreg, pa corr(exchangeable) vce(robust)`.

Another way to obtain potentially more efficient estimates is to introduce a random cluster-specific intercept, so $E(y_{ig}|\mathbf{x}_{ig}, \alpha_g) = \exp(\alpha_g + \mathbf{x}'_{ig}\boldsymbol{\beta})$, where α_g is an independent and identically distributed error. For the Poisson, a closed-form solution arises if $\exp(\alpha_g)$ is gamma distributed. Then one can use the `xtpoisson, re`

command. Alternatively, α_g may be normally distributed, in which case one uses the `xtpoisson, re normal` command. For the NB model, the only option is for α_g to be normally distributed, in which case one uses the `xtnbreg, re` command. Note that for models with an exponential conditional mean, conditioning out the random effect again yields an exponential conditional mean because

$$E(y_{ig} | \mathbf{x}_{ig}) = E\{\exp(\alpha_g)\} \times \exp(\mathbf{x}'_{ig}\boldsymbol{\beta}) = \exp[\ln E\{\exp(\alpha_g)\} + \mathbf{x}'_{ig}\boldsymbol{\beta}],$$
 so only the intercept is changed.

In general, estimator consistency in a random-effects (RE) model requires that the RE distribution be correctly specified. This requirement is not necessary for the Poisson with gamma-distributed random effects, though we need to use the `vce(robust)` option.

The Poisson RE model with normally distributed random effects can be extended to cluster-specific random-slope models using the `mepoisson` command.

FE estimation leads to consistent estimation when there are many observations within each cluster. The simplest approach in a regular count regression is to simply add `i.clid` as regressors, where `clid` denotes the cluster identifier variable. This brute-force approach becomes impractical when there are many clusters. Instead, more creative methods are used. For example, the community-contributed command `poi2hdfe` ([Guimarães 2014](#)) can fit a Poisson model with many fixed effects in two dimensions.

When there are few observations per cluster, consistent estimation of an FE model is generally no longer possible, because of the incidental parameters; see section [22.2.1](#). It is possible for the Poisson model, using the `xtpoisson, fe` command. It is not possible for the NB model—recent literature has led to the view that a model initially proposed as an FE NB model is not really an FE model; see [Guimarães \(2008\)](#). One approach due to [Mundlak \(1978\)](#) is to include the cluster means of regressors as additional regressors; see section 6.6.5 in the linear case.

The same issues arise with panel data. In that case, each individual is a cluster, and the observations for a given cluster are data for each time

period that the individual is observed. A longer discussion for panel data on count outcomes is given in section [22.6](#).

20.9 Quantile regression for count data

Conditional quantile regression (QR) is usually applied to continuous-response data because the quantiles of discrete variables are not unique because the cumulative distribution function is discontinuous with discrete jumps between flat sections. By convention, the lower boundary of the interval defines the quantile in such a case.

Conditional QR for continuous dependent variable has been extended to a special case of a discrete variable model—the count regression. The method, proposed by [Machado and Santos Silva \(2005\)](#), enables QR methods to be applied by suitably smoothing the count data.

20.9.1 Quantile count regression

The key step in the quantile count regression (QCR) model of Machado and Santos Silva is to replace the discrete count outcome y with a continuous variable, $z = h(y)$, where $h(\cdot)$ is a smooth continuous transformation. The standard linear QR methods are then applied to z . Point and interval estimates are then retransformed to the original y scale by using functions that preserve the quantile properties.

The particular transformation used is

$$z = y + u$$

where $u \sim U(0, 1)$ is a pseudorandom draw from the uniform distribution on $(0, 1)$. This step is called “jittering” the count.

Because counts are nonnegative, conventional count models are based on an exponential model for the conditional mean, $\exp(\mathbf{x}'\boldsymbol{\beta})$, rather than a linear function $\mathbf{x}'\boldsymbol{\beta}$. Let $Q_q(y|\mathbf{x})$ and $Q_q(z|\mathbf{x})$ denote the q th quantiles of the conditional distributions of y and z , respectively. Then, to allow for the exponentiation, we specify the conditional quantile for $Q_q(z|\mathbf{x})$ as

$$Q_q(z|\mathbf{x}) = q + \exp(\mathbf{x}'\boldsymbol{\beta}_q) \quad (20.18)$$

The additional term q appears in the equation because $Q_q(z|\mathbf{x})$ is bounded from below by q because of the jittering operation.

To estimate the parameters of a quantile model in the usual linear form $\mathbf{x}'\boldsymbol{\beta}$, we apply a log transformation so that $\ln(z - q)$ is modeled, with the adjustment that if $z - q < 0$, then we use $\ln(\varepsilon)$, where ε is a small positive number. The transformation is justified by the property that, in a correctly specified model, quantiles are equivariant to monotonic transformation (see section 15.2.1) and by the property that quantiles above the censoring point are not affected by censoring from below. Postestimation transformation of the z quantiles back to y quantiles uses the ceiling function, with

$$Q_q(y|\mathbf{x}) = \lceil Q_q(z|\mathbf{x}) - 1 \rceil \quad (20.19)$$

where the symbol $\lceil r \rceil$ in the right-hand side of (20.19) denotes the smallest integer greater than or equal to r . For example, $\lceil 2.4 \rceil = 3$.

To reduce the effect of noise due to jittering, we estimate the parameters of the model multiple times using independent draws from the $U(0, 1)$ distribution and average the multiple estimated coefficients and confidence interval endpoints. Hence, the estimates of the quantiles of y counts are based on $\widehat{Q}_q(y|\mathbf{x}) = \lceil \widehat{Q}_q(z|\mathbf{x}) - 1 \rceil = \lceil q + \exp(\mathbf{x}'\overline{\boldsymbol{\beta}}_q) - 1 \rceil$, where $\overline{\boldsymbol{\beta}}$ denotes the average over the jittered replications.

20.9.2 The qcount command

The QCR method of [Machado and Santos Silva \(2005\)](#) can be performed by using the community-contributed `qcount` command ([Miranda 2006](#)). The command syntax is

```
qcount depvar [ varlist ] [ if ] [ in ], quantile(number) [ repetition(integer) ]
```

where `quantile(number)` specifies the quantile to be estimated and `repetition(integer)` specifies the number of jittered samples to be used to calculate the parameters of the model with the default value being 1,000. The postestimation command `qcount_mfx` computes MES for the model, evaluated at the means of the regressors.

For example, `qcount y x1 x2, q(0.5) rep(500)` estimates a median regression of the count `y` on `x1` and `x2` with 500 repetitions. The subsequent command `qcount_mfx` gives the associated MES.

20.9.3 Doctor-visits data

We illustrate these commands using a dataset on the annual number of doctor visits (`docvis`) by the Medicare elderly in the year 2003. The dataset covers the same individuals as in the doctor-visits dataset used earlier in the chapter, though with nonoverlapping sets of variables.

The regressors used here are an indicator for having private insurance that supplements Medicare (`private`), number of chronic conditions (`totchr`), age in years (`age`), and indicators for `female` and `white`.

We have

```
. * Summary statistics for doctor-visits data
. qui use mus220meps2003qr, clear
. summarize docvis private totchr age female white, separator(0)
```

Variable	Obs	Mean	Std. dev.	Min	Max
docvis	3,677	6.822682	7.394937	0	144
private	3,677	.4966005	.5000564	0	1
totchr	3,677	1.843351	1.350026	0	8
age	3,677	74.24476	6.376638	65	90
female	3,677	.6010335	.4897525	0	1
white	3,677	.9709002	.1681092	0	1

The dependent variable, annual number of doctor visits (`docvis`), is a count. The frequency distribution, not presented, shows that median number of visits is only 5, but there is a long right tail. Around 0.5% of individuals have over 40 visits, and the maximum value is 144.

To demonstrate the smoothing effect of jittering on a count variable, we create the variable `docvisu`, which is obtained for each individual by adding a random uniform variate to the `docvis` variable. We then compare the quantile plot of the smoothed `docvisu` with that for the discrete count `docvis`. For graph readability, values in excess of 40 were dropped. We have

```
.
. * QR: Generate jittered values and compare quantile plots
. set seed 10101
. generate docvisu = docvis + runiform()
. qui qplot docvis if docvis < 40, recast(line) lwidth(medthick)
>      ytitle("Quantiles of doctor visits") saving(graph1, replace)
. qui qplot docvisu if docvis < 40, recast(line) lwidth(medthick)
>      ytitle("Quantiles of jittered doctor visits") saving(graph2, replace)
. graph combine graph1.gph graph2.gph, iscale(1.3) rows(1)
>      ycommon xcommon ysize(2.5) xsize(6)
```

Figure 20.4 shows the step function for the quantile plot of the original count in the first panel and a much smoother quantile plot for the jittered data.

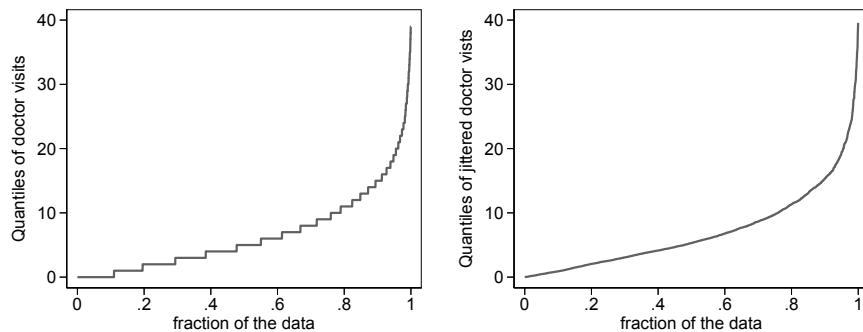


Figure 20.4. Quantile plots of count `docvis` (left) and its jittered transform (right)

20.9.4 Results for NB

The common starting point for regression analysis of counts is Poisson or NB regression. We use the latter and simply print out the MES of a change in the conditional mean of a change in each regressor, evaluated at sample means of the regressors.

```

. * QCR: MEs from conventional negative binomial model
. qui nbreg docvis private totchr age female white, vce(robust)
. margins, dydx(*) atmeans noatlegend
Conditional marginal effects                                         Number of obs = 3,677
Model VCE: Robust
Expression: Predicted number of events, predict()
dy/dx wrt:  private totchr age female white

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
private	1.080582	.2137656	5.05	0.000	.6616087	1.499555
totchr	1.885011	.0771011	24.45	0.000	1.733896	2.036127
age	.0340016	.0176656	1.92	0.054	-.0006224	.0686255
female	-.1398282	.2169893	-0.64	0.519	-.5651194	.2854631
white	.5095404	.6164653	0.83	0.408	-.6987094	1.71779

The MES are computed at the mean, and factor-variable notation is not used to allow direct comparison with the `qcount` command, which does not support factor-variable notation and reports only the MEM.

20.9.5 Results for QCR

We estimate the parameters of the QCR model at the conditional median. We obtain

```

. * QCR for the median
. set seed 10101
. qcount docvis private totchr age female white, q(0.50) rep(500)
> ..... .
> ..... .
> ..... .
> ..... .
> ..... .
> ..... .

Count Data Quantile Regression
( Quantile 0.50 )

Number of obs = 3677
No. jittered samples = 500



|         | Coefficient | Std. err. | z     | P> z  | [95% conf. interval] |
|---------|-------------|-----------|-------|-------|----------------------|
| private | .2025494    | .0410296  | 4.94  | 0.000 | .1221328 .282966     |
| totchr  | .3463796    | .0184165  | 18.81 | 0.000 | .3102838 .3824753    |
| age     | .0083959    | .0033845  | 2.48  | 0.013 | .0017625 .0150294    |
| female  | .0022749    | .0412377  | 0.06  | 0.956 | -.0785495 .0830993   |
| white   | .1195614    | .0997216  | 1.20  | 0.231 | -.0758895 .3150122   |
| _cons   | .0372709    | .2526735  | 0.15  | 0.883 | -.4579601 .5325019   |


```

The statistically significant regressors have the expected signs. The parameters of the model estimated use an exponential functional form for the conditional quantile, so, for example, one more year of aging is associated with an 0.84% increase in the conditional median.

To interpret results, we use the MES, which is easier. The `qcount_mfx` command gives two sets of MEMS after conditional QR. The first is for the jittered variable $Q_q(z|\mathbf{x})$, and the second is for the original count $Q_q(y|\mathbf{x})$. We have

```
. * QCR: MEs after QCR for the median
. qcount_mfx

Marginal effects after qcount
y = Qz(0.50|X)
= 5.05925 (0.0975)
```

	ME	Std. Err.	z	P> z	[95% C.I]	X
private	.92569098	.18616845	4.97	0.0000	0.5608 1.2906	0.50
totchr	1.5792327	.07969824	19.8	0.0000	1.4230 1.7354	1.84
age	.03827918	.01532564	2.5	0.0125	0.0082 0.0683	74.24
female	.01036951	.1879358	.0552	0.9560	-0.3580 0.3787	0.60
white	.51557508	.40793735	1.26	0.2063	-0.2840 1.3151	0.97

```
Marginal effects after qcount
y = Qy(0.50|X)
= 5
```

	ME	[95% C. Set]	X
private	0	0 1	0.50
totchr	1	1 1	1.84
age	0	0 0	74.24
female	0	-1 0	0.60
white	0	-1 1	0.97

The first set of output gives the estimated MEMs for the conditional median $Q_{0.5}(z|x)$ of the jittered variable defined in (20.18). These differ by around 20% from those for the conditional mean aside from a much greater change for the quite statistically insignificant regressor `female`. The standard errors of the MEMs are of similar magnitude. The second set of output gives the estimated MEMs for the conditional quantile of the original discrete count variable $Q_{0.5}(y|x)$ defined in (20.19). These are discretized because the count variable is discrete, and only that for `totchr` differs from zero. We note in passing that if we fit the model using `qreg` rather than `qcount`, then the estimated coefficients are 1 for `private`, 2 for `totchr`, and 0 for the other three regressors, and all standard errors are 0.

QCR at the 75th quantile

The `qcount` command allows one to study the impact of a regressor at different points in the distribution. To explore this point, we reestimate with $q = 0.75$. We have

```

. * QCR: MEs after QCR for q = 0.75
. set seed 10101
. qui qcount docvis private totchr age female white, q(0.75) rep(500)
. qcount_mfx
Marginal effects after qcount
y = Qz(0.75|X)
= 9.06615 (0.1603)

```

	ME	Std. Err.	z	P> z	[95% C.I]	X
private	1.2309624	.33176595	3.71	0.0002	0.5807 1.8812	0.50
totchr	2.3250783	.13360258	17.4	0.0000	2.0632 2.5869	1.84
age	.02662164	.02552073	1.04	0.2969	-0.0234 0.0766	74.24
female	-.0089293	.32846865	-.0272	0.9783	-0.6527 0.6349	0.60
white	1.1806242	.80458387	1.47	0.1423	-0.3964 2.7576	0.97

Marginal effects after qcount

$$y = Qy(0.75|X)$$

$$= 9$$

	ME	[95% C. Set]	X
private	1	0 1	0.50
totchr	2	2 2	1.84
age	0	0 0	74.24
female	0	-1 0	0.60
white	1	-1 2	0.97

For the highly statistically significant regressors, `private` and `totchr`, the MEMS are 30–50% higher than those for the conditional median. As expected, there is less precision at the 75th quantile than at the median, and the standard errors of the MEMs are 60–100% higher.

20.10 Additional resources

The single-equation Stata commands [R] **poisson** and [R] **nbreg** (for `nbreg` and `gnbreg`) cover the basic count regression. See also [R] **poisson postestimation** and [R] **nbreg postestimation** for guidance on testing hypotheses and calculating MES. For zero-inflated and truncated models, see [R] **zip**, [R] **zinb**, [R] **tpoisson**, and [R] **tnbreg**. For Poisson regression with endogenous treatment effects, see [TE] **etpoisson**.

For fitting hurdle models, the community-contributed `hplogit` and `hnblogit` commands are relevant. The community-contributed `prvalue`, `prcounts`, and `countfit` commands are useful for model evaluation and comparison. Stata 15 introduced the `fmm` prefix, which supports finite mixture models of count data as well as variants like zero-inflated models. Estimation of some additional parametric count models using the `gsem` command is presented in section [23.6](#). The community-contributed `qcount` command enables QCR. For clustered and panel count-data analysis, the basic commands [XT] **xtpoisson** and [XT] **xtnbreg** are covered in chapter [22](#). [Deb and Trivedi \(2006\)](#) provide the `mtreatnb` command for estimating the parameters of a treatment-effects model that can be used to analyze the effects of an endogenous multinomial treatment (when one treatment is chosen from a set of more than two choices) on a nonnegative integer-valued outcome modeled using the NB regression.

[Cameron and Trivedi \(2013\)](#) provide a detailed analysis of count regression, and that book's website (<http://cameron.econ.ucdavis.edu/racd2/>) provides additional examples of Stata code for count regression.

20.11 Exercises

1. Consider the Poisson distribution with $\mu = 2$ and a multiplicative mean-preserving lognormal heterogeneity with a variance of 0.25. Using the pseudorandom generators for Poisson and lognormal distributions and following the approach used for generating a simulated sample from the NB2 distribution, generate a draw from the Poisson–lognormal mixture distribution. Following the approach of section [20.2.2](#), generate another sample with a mean-preserving gamma distribution with a variance of 0.25. Using the `summarize`, `detail` command, compare the quantiles of the two samples. Which distribution has a thicker right tail? Repeat this exercise for a count-data regression with the conditional mean function
$$\mu(x) = \exp(1 + 1x),$$
 where x is an exogenous variable generated as a draw from the $\text{uniform}(0, 1)$ distribution.
2. For each regression sample generated in the previous exercise, estimate the parameters of the NB2 model. Compare the goodness of fit of the NB2 model in the two cases. Which of the two datasets is better explained by the NB2 model? Can you explain the outcome?
3. Suppose it is suggested that the use of the `tpoisson` command to estimate the parameters of the ZTP model is unnecessary. Instead, simply subtract 1 from all the counts y , replacing them with $y^* = y - 1$, and then apply the regular Poisson model using the new dependent variable y^* ; $E(y^*) = E(y) - 1$. Using generated data from $\text{Poisson}\{\mu(x) = 1 + x\}$, $x = \text{uniform}(0, 1)$, verify whether this method is equivalent to the `tpoisson` command.
4. Using the finite-mixture prefix (`fmm`), fit two- and three-component NB2 mixture models for the univariate (intercept only) version of the `docvis` model. Use the BIC to select the “better” model. For the selected model, use the `estat lcmean` command to compute the means of the m components. Explain and interpret the estimates of the component means and the estimates of the mixing fractions. Is the identification of the two and three components robust? Explain your answer.
5. Continuing with the previous example of a two-component mixture, use the command `estat lcprob` to generate the latent class posterior

probability distributions. Use histograms to compare the distributions. Verify that the means of these two component distributions equal mixture proportions, respectively.

6. For this exercise, use the data from section [20.6](#). Estimate the parameters of the Poisson and ZIP models using the same covariates as in section [20.6](#). Test whether there is statistically significant improvement in the log likelihood. Which model has a better BIC? Contrast this outcome with that for the NB2/ZINB pair and rationalize the outcome.
7. Consider the data application in section [20.7.2](#). Drop all observations for which the `medicaid` variable equals one, and therefore also drop `medicaid` as a covariate in the regression. For this reduced sample, estimate the parameters of the Poisson model, treating the `private` variable first as exogenous and then as endogenous. Obtain and compare the two estimates of the ME of `private` on `docvis`. Implement the test for endogeneity given in section [20.7.2](#).
8. Use the data from section [20.9](#), except let the dependent count variable be `totchr`, and drop `totchr` from the regressors. Using the community-contributed `qcount` command, estimate `qcount` regressions for $q = 0.25, 0.50$, and 0.75 , and use `qcount_mfx` to calculate the MES. Store and print the results in tabular form. Explain how this command works in the case of `qcount` regressions and whether there are any differences in the interpretation compared with the standard Poisson regression.

Chapter 21

Survival analysis for duration data

21.1 Introduction

Duration data or survival data or failure data are data on a variable that measures the length of time spent in a state before transition to another state. An economics example is the length of time spent unemployed before transition to a job or withdrawal from the workforce. A biomedical example is the number of years a person survives after joining a medical study. A quality control example is the number of hours a light bulb functions before failure.

If spells are completely observed, then the regression methods of the previous chapters are directly applicable. A conditional mean approach specifies an exponential form for the conditional mean because durations are strictly positive. From stochastic process theory, a natural starting point is the exponential distribution. There is no explicit exponential estimation command in Stata. But exponential regression falls in the generalized linear models class, and the command `glm y x, family(gamma) scale(1) link(log) vce(robust)` for duration data on completed spells is analogous to the command `poisson y x, vce(robust)` for count data. A fully parametric approach considers more flexible distributions than the exponential, such as the Weibull and Gompertz distributions. This is analogous to using the negative binomial distribution rather than the Poisson for count data.

A major complication for duration data is that a range of commonly used sampling schemes leads to the spell length being incompletely observed, so the dependent variable is censored. For example, in biostatistics analysis of survival following treatment or onset of a medical condition, if individuals are followed for five years, then not all spells are complete if some individuals survive more than five years or if some individuals leave the study. A fully parametric approach, analogous to a tobit model with right-censoring, has the limitation of heavy reliance on distributional assumptions.

Instead, the biostatistics literature emphasizes a semiparametric approach, the Cox proportional hazards (PH) model, that is uniquely

applicable to duration data. This approach brings in terms and concepts not used in the analysis of other types of data, such as the survivor function, hazard function, and cumulative hazard function. The methods are sufficiently specialized that Stata provides a separate set of commands that begin with the letters `st` (for survival time).

Additionally, an important issue that often arises in microeconomics applications with duration data is distinguishing between the roles of individual heterogeneity and duration dependence in determining spell length. For example, we may observe that individuals with longer unemployment spells are less likely to exit unemployment in the next period. One reason for this phenomenon is individual heterogeneity in ability to be reemployed because of individual variation in, for example, job skills. An alternative reason is that being unemployed for a long time by itself harms the probability of reemployment. These two explanations have quite different policy implications but are difficult to disentangle.

This chapter provides an introductory treatment of these special features of duration data analysis for the simplest case of randomly censored duration data where only one spell is observed per individual and, in most examples, individual regressors are constant throughout the spell. We begin with the popular Cox PH model, which is a semiparametric model that focuses on the role of individual observed heterogeneity. We then present maximum likelihood (ML) estimation of a range of parametric regression models that differ in their specification of the roles of observed individual heterogeneity and duration dependence. The chapter concludes with the discrete-time hazards binary outcome model that, like the Cox model, relies on weaker assumptions than ML estimation.

21.2 Data and data summary

The example for this chapter models the length of time without a job, where individuals are observed at the beginning of the spell but are not necessarily observed for the full length of the spell.

21.2.1 Summary statistics

The data on the length of jobless spells are those analyzed in [Cameron and Trivedi \(2005, 603–608\)](#) and originally analyzed by [McCall \(1996\)](#), who provides complete details.

The dependent variable `spell` is the number of periods jobless (measured in two-week intervals).

The spells are censored from above. A spell is considered to be complete when a person moves to a full-time job; otherwise, the spell is censored. The binary variable `censor1` equals 1 if a person becomes reemployed at a full-time job (so is uncensored) and equals 0 (so is censored) otherwise.

The dataset includes many regressors. We focus on two regressors, an indicator variable for whether the person filed an unemployment insurance claim (variable `ui`) and log weekly earnings before the jobless spell (variable `logwage`). These regressors do not vary through the spell.

We begin with variable description and summary statistics.

```
. * Unemployment spell length: Describe and summarize key variables
. qui use mus221mccall, clear
. describe spell censor1 ui logwage
```

Variable name	Storage type	Display format	Value label	Variable label	
spell	int	%8.0g		Periods jobless: two-week intervals	
censor1	int	%8.0g		Reemployed at full-time job	
ui	int	%8.0g		Filed UI claim	
logwage	float	%8.0g		log weekly earnings	
. summarize spell censor1 ui logwage					
Variable	Obs	Mean	Std. dev.	Min	Max
spell	3,343	6.247981	5.611271	1	28
censor1	3,343	.3209692	.4669188	0	1
ui	3,343	.5527969	.4972791	0	1
logwage	3,343	5.692994	.5356591	2.70805	7.600402

There are observations on 3,343 individuals. Only 32% of spells are observed to completion; the remaining spells are censored. The mean spell length, averaged over individuals with either complete or incomplete spells, is 6.2 periods. This most likely understates the mean length of completed spells because so many spells are not observed to completion. A bit more than one-half of the individuals filed an unemployment insurance claim.

It is helpful to list the first few observations.

```
. * List the first six observations of key variables
. list spell censor1 ui logwage in 1/6, clean
```

	spell	censor1	ui	logwage
1.	5	1	0	6.89568
2.	13	1	1	5.28827
3.	21	1	1	6.76734
4.	3	1	1	5.97889
5.	9	0	1	6.31536
6.	11	0	1	6.85435

The first four spells were complete and of lengths, respectively, 5, 13, 21, and 3 periods. The 5th and 6th spells were incomplete, so they lasted at least 9 and 11 periods, respectively.

21.2.2 The stset command

Stata commands for duration analysis begin with `st` and are called `st` (survival-time) commands. The `help st` command provides a complete list of the various commands.

To use these commands, one must first declare the data to be survival data using the `stset` command. At a minimum, the dependent variable needs to be identified. Additionally, if data are censored, the censoring variable needs to be identified.

Data may be single-record data, with one observation per individual, or multiple-record data, with more than one observation per individual. And data may be single-failure data, with at most one failure per record, or multiple-failure data. The current example is one of single-record, single-failure data.

For single-spell data, the `stset` command has syntax

```
stset timevar [if] [weight] [, options]
```

The main option is the `failure()` option, which declares the censoring variable, if there is censoring. The `st` commands call the completion of a spell a failure. This uses the language of biostatistics, where a spell may end with death, or of quality control, where a spell may end with a light bulb blowing. In many economics contexts, such as the current one, the ending of a spell is actually a success.

In the simplest case, the spell begins, and the subject is at risk at time 0. The `origin()` option defines when a subject becomes at risk, the `enter()` option specifies when a subject first enters the study, and the `exit()` option specifies when a subject exits the study.

Once the dependent variable and, if applicable, the censoring variable are declared, there is no need to include them in subsequent commands. For example, for parametric regression of `y` on `x` with censored data, the command is `streg x` rather than `reg y x`.

For the current data, we have

```

. * Command stset defines the dependent and censoring variables
. stset spell, fail(censor1=1)
Survival-time data settings
    Failure event: censor1==1
Observed time interval: (0, spell]
    Exit on or before: failure


---


    3,343  total observations
        0  exclusions


---


    3,343  observations remaining, representing
    1,073  failures in single-record/single-failure data
    20,887  total analysis time at risk and under observation
                    At risk from t =      0
                    Earliest observed entry t =  0
                    Last observed exit t =   28

```

The option `fail(censor1=1)` means that observations on `spell` will be viewed as uncensored (a failure) if `censor1` equals one and otherwise will be viewed as censored. Note that missing values of `censor1` will then be considered censored.

There are 1,073 complete spells (a “failure” in Stata terminology) out of 3,343, so a proportion $1073/3343 = 0.321$ of spells are complete, as was given in the original summary statistics. The total analysis time at risk is the sum over individuals of the spell length for each individual. This equals the number of individuals times the average spell length, here $3343 \times 6.248 = 20887$.

21.2.3 Survival data organization

The `stdescribe` command can be used to obtain additional information.

```
. * Survival description of dataset
. stdescribe
    Failure _d: censor1==1
Analysis time _t: spell
```

Category	Total	Per subject			
		Mean	Min	Median	Max
Number of subjects	3343				
Number of records	3343	1	1	1	1
Entry time (first)		0	0	0	0
Exit time (final)		6.247981	1	5	28
Subjects with gap	0				
Time on gap	0				
Time at risk	20887	6.247981	1	5	28
Failures	1073	.3209692	0	0	1

There is exactly one spell for each individual. There are no gaps in time in the data. The spell lengths range from 1 period to 28 periods

The `st` commands allow multiple spells per individual. In this introductory treatment, we consider only single-spell data.

The `stvary` command shows whether regressors vary during the spell and are missing in some spells. This is applicable for use with multirecord data. We illustrate the command, even though here the data are single-spell data.

```
. * Variation in regressors over time - relevant for multiple-record data
. stvary ui logwage
    Failure _d: censor1==1
Analysis time _t: spell
```

Variable	Subjects for whom the variable is			never	always	sometimes
	constant	varying		missing	missing	missing
ui	3343	0		3343	0	0
	3343	0		3343	0	0

There are no missing values for the two regressors.

The `stsum` command provides summary statistics for the dependent variable. Here we use the `by()` option to additionally provide these statistics by unemployment insurance status.

```

. * Summary of survival data by insurance status
. stsum, by(ui)

      Failure _d: censor1==1
      Analysis time _t: spell

```

ui	Time at risk	Incidence rate	Number of subjects	Survival time		
				25%	50%	75%
0	6,135	.0938875	1495	2	9	.
1	14,752	.0336903	1848	9	20	.
Total	20,887	.0513717	3343	5	15	.

Considering the final total row, we see there were 20,887 periods at risk. From the `stdescribe` command, there were 1,073 failures, so the incidence rate is $1073/20887 = 0.051$. On average in each period, 5.1% of the ongoing spells resulted in failure.

The median survival time of 15 periods is computed using methods detailed in the next section. Because of the extent of censoring and consequent incomplete spells, the 75th percentile of survival time cannot be computed.

Comparing these statistics by unemployment insurance status, we clearly see that individuals who filed an unemployment insurance claim (`ui=1`) are much less likely to exit to a full-time job.

21.3 Survivor and hazard functions

The graphs commonly used to display the distribution of a single variable are histograms and kernel density estimates. These are not as useful when the distribution is not completely observed because of censoring, here censoring from above.

Instead, the standard graphs for censored duration data are graphs of the survivor function, cumulative hazard function, and hazard function. These are presented in this section.

21.3.1 Densities for complete and incomplete spells

For completeness, we begin with histograms and kernel density estimates for the dependent variable, partitioned by whether spells are complete or incomplete.

```
. * Graph histogram and density of survival data by ui status
. qui graph twoway (hist spell if censor1==1)
>      (kdensity spell if censor1==1, lwidth(thick) lstyle(p1)),
>      legend(pos(1) col(1) ring(0)) title("Completed spells")
. qui graph twoway (hist spell if censor1==0)
>      (kdensity spell if censor1==0, lwidth(thick) lstyle(p1)),
>      legend(pos(1) col(1) ring(0)) title("Incomplete spells")
```

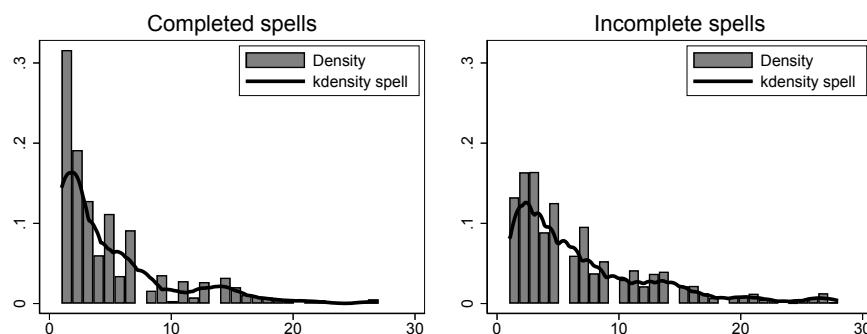


Figure 21.1. Histograms for complete and incomplete spells

Figure 21.1 presents the graph. Shorter spells are more common than longer spells and, compared with the incomplete spells, the completed spells show a greater proportion of spells that are short.

21.3.2 Survivor function

The standard notation in the survival literature is to denote the dependent random variable by T , for time, rather than the usual Y . The survivor function is

$$S(t) = \Pr(T > t)$$

and is equal to one minus the cumulative distribution function (c.d.f.).

The standard estimate of $S(t)$ is the Kaplan–Meier nonparametric estimate of the survivor function. Command `sts list` provides and lists this estimate of the survivor function.

```

. * Compute survivor function
. sts list
      Failure _d: censor1==1
      Analysis time _t: spell

```

Kaplan-Meier survivor function

Time	At risk	Fail	Lost	Survivor function	Std. error	[95% conf. int.]
1	3343	294	246	0.9121	0.0049	0.9019 0.9212
2	2803	178	304	0.8541	0.0062	0.8415 0.8659
3	2321	119	305	0.8103	0.0071	0.7960 0.8238
4	1897	56	165	0.7864	0.0076	0.7712 0.8008
5	1676	104	233	0.7376	0.0085	0.7206 0.7538
6	1339	32	111	0.7200	0.0088	0.7023 0.7369
7	1196	85	178	0.6688	0.0098	0.6492 0.6876
8	933	15	70	0.6581	0.0100	0.6380 0.6773
9	848	33	98	0.6325	0.0106	0.6113 0.6528
10	717	3	55	0.6298	0.0106	0.6086 0.6503
11	659	26	77	0.6050	0.0113	0.5825 0.6267
12	556	7	40	0.5974	0.0115	0.5744 0.6195
13	509	25	69	0.5680	0.0123	0.5434 0.5918
14	415	30	74	0.5270	0.0135	0.5001 0.5531
15	311	19	40	0.4948	0.0146	0.4658 0.5230
16	252	10	41	0.4751	0.0153	0.4449 0.5047
17	201	8	24	0.4562	0.0161	0.4245 0.4874
18	169	7	13	0.4373	0.0169	0.4040 0.4702
19	149	4	15	0.4256	0.0174	0.3912 0.4595
20	130	3	18	0.4158	0.0179	0.3804 0.4507
21	109	4	23	0.4005	0.0188	0.3635 0.4372
22	82	4	9	0.3810	0.0203	0.3412 0.4206
23	69	0	9	0.3810	0.0203	0.3412 0.4206
24	60	0	2	0.3810	0.0203	0.3412 0.4206
25	58	0	10	0.3810	0.0203	0.3412 0.4206
26	48	2	13	0.3651	0.0223	0.3214 0.4088
27	33	5	24	0.3098	0.0296	0.2528 0.3684
28	4	0	4	0.3098	0.0296	0.2528 0.3684

The survivor function estimate is computed as follows. At time 1, 294 out of 3343 observations failed, so a proportion $294/3343 = 0.0879$ failed and $1 - 0.0879 = 0.9121$ survived. At time 2, there were 2803 observations at risk because 294 observations were lost at time 1 because of failure and 246 were lost because of random censoring. Of these 2,803 at-risk observations, 178 then failed, so at time 2 a proportion $178/2803 = 0.0635$

failed and $1 - 0.0635 = 0.9365$ survived. Cumulatively $0.9121 \times 0.9365 = 0.8541$ survived to the end of period 2, and so on.

The precise formula for the Kaplan–Meier estimate of the survivor function is

$$\widehat{S}(t) = \prod_{j|t_j \leq t} \left(\frac{n_j - d_j}{n_j} \right)$$

where $t_j, j = 1, \dots$ denotes the times at which failures occur, n_j denotes the number of individuals at risk of failure just before time t_j , and d_j denotes the number of failures at time t_j . In the current example, failures occur at consecutive time periods 1, 2, ..., 28, but the methods are applicable to the more general case where failures may not occur in all time periods.

Command `sts graph` provides a plot of these estimates. We produce this graph with 95% pointwise confidence bands, along with a similar graph segmented by unemployment insurance status.

```
. * Graph survivor function over all and by ui
. qui sts graph, survival ci legend(pos(8) col(1) ring(0))
. qui sts graph, by(ui) survival ci legend(pos(8) col(1) ring(0))
```

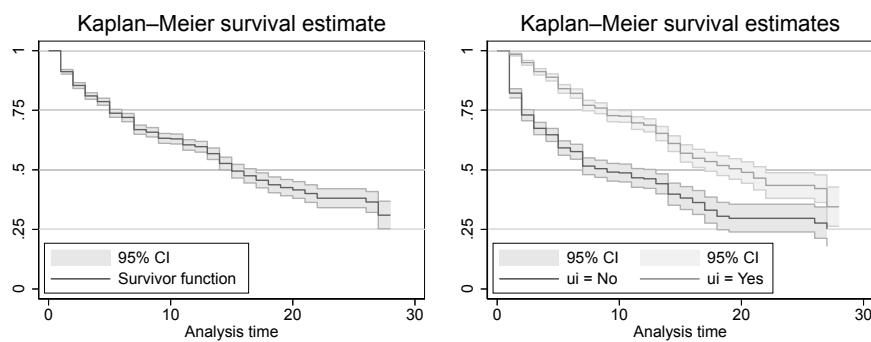


Figure 21.2. Kaplan–Meier estimate of survivor function

Figure 21.2 presents the graph. The survivor function can be estimated to time $t = 28$, by which time the probability of remaining without a full-time job is estimated to be around 0.3. The survivor curve is higher for those who filed an unemployment insurance claim ($u_{i1}=1$).

21.3.3 Hazard rate and cumulative hazard function

The hazard rate is the instantaneous probability of the spell ending at time t , conditional on survival to time t . In the current example, the hazard rate at time $t = 8$, for example, is the probability of finding a full-time job in the 8th period, given that a full-time job was not found in the first 7 periods.

Let $f(t)$ denote the density of T , let $F(t) = \Pr(T \leq t)$ denote the c.d.f., and, as before, let $S(t) = \Pr(T > t) = 1 - F(t)$ denote the survivor function. The hazard rate $h(t)$ is defined to equal $f(t)$, the instantaneous probability of the spell ending, divided by $S(t)$, the probability of the spell lasting to time t . The hazard function or hazard rate is then

$$h(t) = \lim_{h \rightarrow \infty} \frac{\Pr(t \leq T < t + h) | T > t)}{h} = \frac{f(t)}{S(t)} = \frac{f(t)}{1 - F(t)} \quad (21.1)$$

An advantage of modeling $h(t)$ is that, for censored data, it can be nonparametrically estimated up to t_{\max} , the maximum observed length of a completed spell in the dataset. This does not allow estimation of the mean spell length if data are censored. But it is still very informative because the hazard rate is of interest per se. In the current example, a hazard rate that is decreasing in t implies a serious policy problem because the longer a person remains without a full-time job, the less likely he or she is to obtain one.

A simple estimate of the hazard function is

$$\hat{h}(t) = \frac{d_j}{n_j} \quad (21.2)$$

the number of failures at time t divided by the number at risk immediately before time t . For example, from the preceding output at $t = 5$, there were 1,676 observations at risk and 104 failed, so $\hat{h}(5) = 104/1676 = 0.0605$.

This estimate is exceptionally noisy because it is based on only a small proportion of the n_j observations. It is common to instead produce a kernel-smoothed version of the hazard function. This can be obtained using the `hazard` option of the `sts graph` command.

A related measure is the cumulative hazard function $H(t) = -\ln S(t)$, with

$$H(t) = \sum_{j|t_j \leq t} h(t_j)$$

which is estimated by the Nelson–Aalen estimate of the cumulative hazard function.

$$\hat{H}(t) = \sum_{j|t_j \leq t} \hat{h}(t_j) = \sum_{j|t_j \leq t} \frac{d_j}{n_j}$$

The following code provides estimates of the cumulative hazard function and the hazard function, along with 95% confidence bands.

```
. * Graph cumulative hazard function and smoothed hazard function
. local endash = ustrunescape("\u2013")
. qui sts graph, cumhaz ci legend(pos(11) col(1) ring(0))
>      title("Nelson`endash`Aalen cumulative hazard")
. qui sts graph, hazard ci legend(pos(8) col(1) ring(0))
```

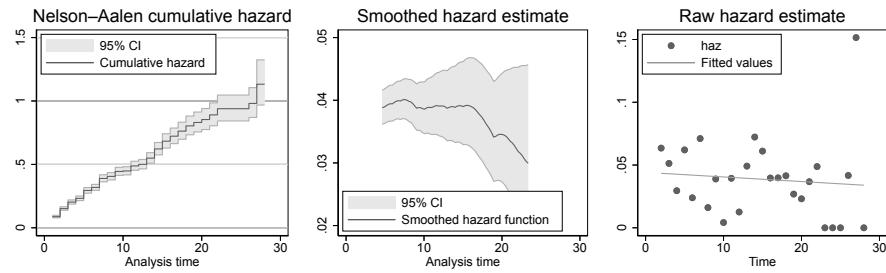


Figure 21.3. Kaplan–Meier estimate of survivor function

From the second panel of figure 21.3, the hazard function is very noisily estimated, even after smoothing. The hazard appears to be declining, especially for $t > 16$.

The third panel of figure 21.3 presents an estimate of the raw hazard function before smoothing. This was obtained using the following commands.

```
. * Graph raw hazard rate
. qui sts list, cumhaz saving(cumhazard, replace)
. preserve
. use cumhazard, clear
. qui generate haz = cumhaz - cumhaz[_n-1]
. qui graph twoway (scatter haz time) (lfit haz time),
>     legend(pos(11) col(1) ring(0)) title("Raw hazard estimate")
. restore
```

21.4 Semiparametric regression model

We now consider regression, beginning with the Cox PH regression model, which does not require complete specification of the conditional density of the data. This model focuses on estimating the role of individual observed heterogeneity while controlling for duration dependence in a flexible way.

The model is especially useful in clinical trials that measure the effect on the hazard rate of taking a drug because coefficient estimates can be interpreted as, for example, taking the drug decreases the hazard of death by 10%.

The model is less useful if interest lies in the nature of duration dependence over time because its flexibility often leads to imprecise estimation of the time-varying portion of the hazard.

21.4.1 The `stcox` command

Estimates of the Cox PH regression model are obtained using the `stcox` command, which has syntax

```
stcox [indepvars] [if] [in] [, options]
```

An important option is the `nohr` option, which, as explained in the next subsection reports raw coefficients rather than exponentiated coefficients. Note that *varlist* is just a list of the regressors. It is not necessary to provide the names of the dependent variable and any censoring variable because these are already declared in the `stset` command.

The postestimation commands `predict`, `margins`, `stcurve`, `stphplot`, `stcoxkm`, and `estat phtest` are presented in this section.

21.4.2 Cox proportional hazards model

The Cox PH model defines the hazard rate for dependent variable t , conditional on regressors \mathbf{x} , for the i th individual to be

$$h(t_i | \mathbf{x}_i) = h_0(t_i) \times \exp(\mathbf{x}'_i \boldsymbol{\beta}) \quad (21.3)$$

The term “proportional hazards” is used because the model restricts the hazard rate to be the same baseline hazard rate function $h_0(\cdot)$ for all individuals. This baseline hazard is then scaled up or down for each individual by $\exp(\mathbf{x}'_i \boldsymbol{\beta})$, called the relative hazard. In the simplest case, given here, the regressors \mathbf{x}_i are constant through an individual’s spell. This assumption can be relaxed to allow time-varying regressors; see section [21.8](#).

The maximum partial-likelihood estimation method for the Cox PH model is detailed in, for example, [Cameron and Trivedi \(2005, 594–596\)](#). A major feature is that the method is robust to independent censoring, meaning that after conditioning on any regressors, the censoring mechanism is independent of the duration process for the dependent variable t . For example, there is no problem if we randomly lost track of someone in the sample or if we simply have not observed them to the end of their spell.

Estimation yields estimates of the parameters $\boldsymbol{\beta}$. Using calculus methods, we see a one-unit change in the j th regressor is associated with the following change in the hazard rate:

$$\frac{\partial h(t_i | \mathbf{x}_i)}{\partial x_{ij}} = h_0(t_i) \times \exp(\mathbf{x}'_i \boldsymbol{\beta}) \times \beta_j = h(t_i | \mathbf{x}_i) \times \beta_j$$

It follows that if $\beta_j > 0$, then an increase in the regressor is associated with an increase in the hazard rate and hence in the likelihood of failure. For example, it may be that the higher the wage in the previous job, the more likely it is that we will exit from unemployment given the current length of the unemployment spell. And if $\beta_j = 2\beta_k$, for example, then the effect on the hazard rate of changing x_j is twice that of changing x_k .

For censored data, the Cox PH model cannot produce estimates of the conditional mean. It provides only estimates of the hazard function and related functions such as the survivor function. For duration data, the hazard function is of intrinsic interest. In principle, a similar estimator can be applied to right-censored expenditure data, in lieu of a tobit model. But for data such

as expenditure data, the hazard function, the ratio of the conditional density to one minus the conditional c.d.f., is of no interest.

21.4.3 Cox PH estimates

Estimation using the `stcox` command with the `nohr` option yields

Cox regression with Breslow method for ties					
			Number of obs = 3,343		
No. of subjects = 3,343					
No. of failures = 1,073					
Time at risk = 20,887			Wald chi2(2) = 287.10		
Log pseudolikelihood = -7847.2989			Prob > chi2 = 0.0000		
_t	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
ui	-1.007489	.0609082	-16.54	0.000	-1.126866 -.8881108
logwage	.3991956	.0525997	7.59	0.000	.296102 .5022892

The hazard rate increases with higher wage in the previous job (variable `logwage`) and decreases for those who have filed for unemployment insurance benefits (`ui=1`). Furthermore, these effects are highly statistically significant.

The magnitude of these effects is not immediately clear. Consider a one-unit change in regressor x_1 where we have partitioned $\mathbf{x}'\beta = \exp(\beta_1 x_1 + \mathbf{x}'_2 \beta_2)$. Then the hazard rate defined in (21.3) is e^{β_1} times larger because

$$\frac{h_0(t_i) \exp\{\beta_1(x_{1i} + 1) + \mathbf{x}'_{2i}\beta_2\}}{h_0(t_i) \exp(\beta_1 x_{1i} + \mathbf{x}'_{2i}\beta_2)} = \frac{\exp(\beta_1 x_{1i} + \mathbf{x}'_{2i}\beta_2) \times \exp(\beta_1)}{\exp(\beta_1 x_{1i} + \mathbf{x}'_{2i}\beta_2)} = \exp(\beta_1)$$

The default form of the `stcox` command reports these interpretable exponentiated coefficients $e^{\hat{\beta}}$, rather than $\hat{\beta}$. Using the default version of the `stcox` command, we obtain

. * Cox PH regression with exponentiated coefficients						
. stcox ui logwage, vce(robust) nolog						
Failure _d: censor1==1						
Analysis time _t: spell						
Cox regression with Breslow method for ties						
No. of subjects = 3,343				Number of obs = 3,343		
No. of failures = 1,073						
Time at risk = 20,887						
				Wald chi2(2) = 287.10		
Log pseudolikelihood = -7847.2989				Prob > chi2 = 0.0000		
<hr/>						
_t	Haz. ratio	Robust std. err.	z	P> z	[95% conf. interval]	
ui	.3651348	.0222397	-16.54	0.000	.3240471	.4114323
logwage	1.490625	.0784065	7.59	0.000	1.344607	1.6525

The underlying estimates are the same as those obtained with the option `nohr`, and the fitted log likelihood is the same. The only difference is that the coefficients are now reported as hazard ratios, leading to different standard errors and 95% confidence interval. The hazard ratio for variable `ui` is the exponential of the coefficient previously reported because $\exp(-1.007489) = 0.365135$. Note that the z statistics and p -values are the same as those given in the preceding output and are interpreted as tests against an exponentiated coefficient taking value 1.

The model results are interpreted as follows. Filing for unemployment insurance benefits is associated with the hazard of the unemployment spell ending being only 0.365 times what it would be otherwise, a substantial decrease. Similarly, a one-unit change in `logwage` leads to the hazard rate being 1.491 times higher. These economic variables clearly have a very large effect on the duration of the unemployment spell.

The estimation results are usually interpreted in this way; marginal effects are rarely computed.

21.4.4 Prediction and marginal effects

As already noted, the Cox PH estimator controls for censoring using minimal assumptions that do not produce estimates of the conditional mean. Instead the hazard function is identified. The default `hr` option of the `predict`

postestimation command predicts the relative hazard $\exp(\mathbf{x}'\boldsymbol{\beta})$, while the `xb` option predicts $\mathbf{x}'\boldsymbol{\beta}$.

Other `predict` options produce predictions only for uncensored observations with completed spells. The options `basehc`, `basesurv`, and `basechazard` produce, respectively, the baseline hazard contributions, baseline survivor function, and baseline cumulative hazard. Various residuals can be obtained: martingale-like (`mgale`), Cox–Snell (`csnell`), deviance (`deviance`), efficient scores (`scores`), Schoenfeld (`schoenfeld`), and scaled Schoenfeld (`scaledsch`); see [ST] **stcox postestimation**.

The only options of the `margins` command, aside from `predict (statistic)`, are `hr` and `xb`. The default option is `hr`. The `margins dydx (*)` command, used extensively elsewhere in this book, is of no use here because it gives marginal effects (MEs) with respect to the relative hazard $\exp(\mathbf{x}'\boldsymbol{\beta})$, rather than the hazard rate $h_0(t) \times \exp(\mathbf{x}'\boldsymbol{\beta})$. Instead, we interpret the exponentiated coefficients as relative hazard rates.

21.4.5 The `stcurve` command

The `stcurve` command provides graphs of the survivor, cumulative hazard, and hazard functions following several `st` estimation commands, including `stcox` and `streg`. For `stcox`, the graphs are produced only for uncensored observations.

These functions of time, such as the hazard $h(t|\mathbf{x}) = h_0(t) \times \exp(\mathbf{x}'\boldsymbol{\beta})$, vary according to the regressor values. The Stata default is to evaluate at the mean value of the regressors. The `at (varname=#)` option instead evaluates at values of specified regressors and at the mean of other regressors.

Here we evaluate at `ui=1` and at the mean of `logwage`.

```
. * PH model curves: Survivor, cumulative hazard, and hazard functions
. qui stcurve, survival at(ui=1) title("Survivor function")
. qui stcurve, cumhaz at(ui=1) title("Cumulative hazard function")
. qui stcurve, hazard at(ui=1) title("Hazard function")
```

Figure 21.5 presents the graphs. They have similar shape to those obtained earlier for variable `spell` in the nonregression case. Bootstrap

confidence intervals for the survival curve can be obtained using the community-contributed `bsurvci` command ([Ruhe 2019](#)). The article provides Stata code that can be adapted to other quantities computed by the `predict` postestimation command following the `stcox` or `streg` commands.

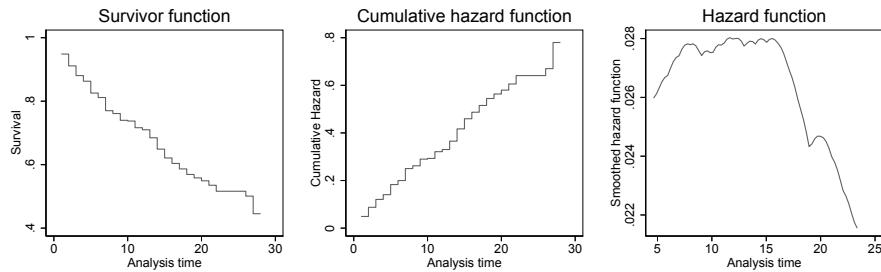


Figure 21.4. Survivor and related curves following Cox PH regression

21.4.6 Diagnostics for the PH model

The Cox PH model has the attraction of not being fully parametric, but it does make the key assumption that the hazard rate $h(t|\mathbf{x})$, which depends on both duration (t) and regressors (\mathbf{x}), factorizes into two separate components, so $h(t|\mathbf{x}) = h_0(t) \times \exp(\mathbf{x}'\boldsymbol{\beta})$.

Stata provides three diagnostics for the Cox PH model: `stphplot`, `stcoxkm`, and `estat phtest`.

PH imply proportional integrated hazards, with $H(t|\mathbf{x}) = H_0(t) \times \exp(\mathbf{x}'\boldsymbol{\beta})$, where $H(t) = \int_0^t h(s)ds$. Taking the logarithm, $\ln H(t|\mathbf{x}) = \ln H_0(t) + \mathbf{x}'\boldsymbol{\beta}$. So the log-integrated hazard curves, also called the log–log survivor curves, should be parallel at different values of the regressors.

We use the `stphplot` command to contrast the log–log survivor curves at the two different values of `ui`, controlling for `logwage`, and at high and low wages (using newly created variable `highwage`), controlling for `ui`.

```

. * PH model diagnostics: Check for parallel log-log survival curves
. qui stphplot, by(ui) adjust(logwage) legend(pos(1) col(1) ring(0))
>      title("Log`endash`log survival by UI")
. qui summarize logwage, d
. qui generate highwage = logwage > r(p50)
. qui stphplot, by(highwage) adjust(ui) legend(pos(1) col(1) ring(0))
>      title("Log`endash`log survival by wage")

```

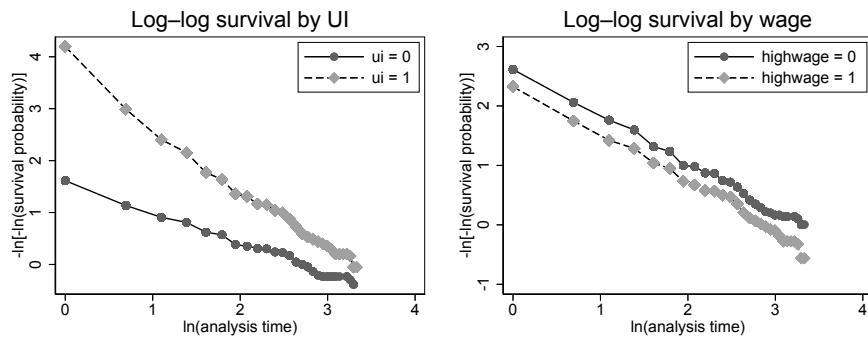


Figure 21.5. Check for parallel log–log survivor curves

Figure 21.5 presents the log–log survivor curves. The curves are parallel for low and high wages, but for `ui` they are not close to parallel, so the PH assumption does not appear to be appropriate. The recommendation would be to fit separate Cox PH models for the two levels of `ui`.

The `stcoxkm` command plots, for each level of the regressor, both the fitted survivor function from Cox regression and the (nonregression) Kaplan–Meier estimate of the survivor function. The two survivor curves should be similar if the PH is a good model. We do this for both `ui` and for `highwage`.

```

. * PH model diagnostics: Check for good prediction of survival curve
. qui stcoxkm, by(ui) legend(pos(1) col(1) ring(0))
>      title("Survival predicted by UI")
. qui stcoxkm, by(highwage) legend(pos(1) col(1) ring(0))
>      title("Survival predicted by wage")

```

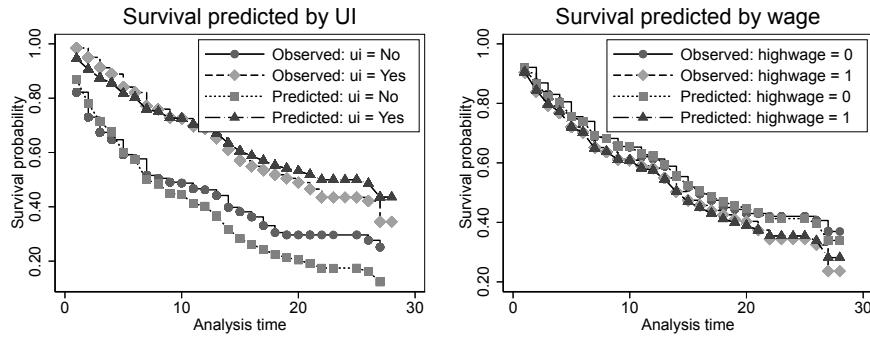


Figure 21.6. Compare Cox PH to Kaplan–Meier survivor curve

From figure 21.6, the model is reasonable for `highwage` but does not do so well for `ui`.

The final test is a formal statistical test that, unlike the previous graphical diagnostics, can be applied to regressors that take many values. These tests are based on a quantity computed for each observation and for each regressor called the scaled Schoenfeld residual. These residuals should be independent of spell length or any function of spell length.

This test is implemented using the `estat phtest` command. We use the default option of testing whether the residuals are related to spell length. We obtain

```
. * PH model diagnostics: Test of PH assumption
. qui stcox ui logwage, vce(robust) nolog
. estat phtest, detail
```

Test of proportional-hazards assumption

Time function: Analysis time

	rho	chi2	df	Prob>chi2
ui	0.26267	61.73	1	0.0000
	-0.06872	4.14	1	0.0419
Global test		61.75	2	0.0000

Note: Robust variance-covariance matrix used.

There is clearly a problem with variable `ui`, while the null hypothesis of no relationship with spell length is just rejected for variable `logwage` at

significance level 0.05.

The following code presents a visual version of this test by obtaining the scaled Schoenfeld residuals and plotting them against spell length.

```
. * PH model diagnostics: Graph the Schoenfeld residual against time  
. qui stcox ui logwage, vce(robust) nolog  
. qui predict double rui rlogwage, schoenfeld  
. graph twoway (scatter rui _t) (lfit rui _t), legend(off)  
> ytitle("Schoenfeld residual for ui") legend(off)  
. graph twoway (scatter rlogwage _t) (lfit rlogwage _t), legend(off)  
> ytitle("Schoenfeld residual for logwage")
```

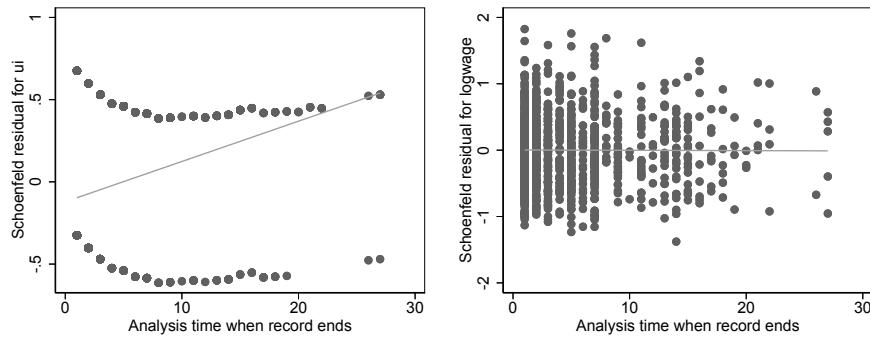


Figure 21.7. Graph of Schoenfeld residuals against spell length

Figure 21.7 presents the resulting graph. It is clear that for the residuals corresponding to variable `ui`, there is a positive relationship with spell length, while there appears to be no relationship between `logwage` and the spell length. Again, the Cox PH assumption is inappropriate for variable `ui`.

The manual entry [ST] **stcox postestimation** presents further details on diagnostic methods, many of which can also be applied after command `streg`.

21.4.7 Competing risks models

The analysis has assumed that the only way to exit from unemployment is to obtain a full-time job. In fact, people also exit to other jobs such as a part-time job. These other reasons for exiting are known as competing risks. A

brief presentation is given here; see [Cleves, Gould, and Marchenko \(2016\)](#), for example, for a more complete analysis.

These competing risks have been treated as independent censoring, so that in (21.2) the estimated hazard rate $\hat{h}(t) = d_j/n_j$ has a risk set n_j that does not include persons who exit due to obtaining a job that is not full-time. This smaller value of n_j leads to overestimation of the hazard of finding full employment because those who have found other employment such as part-time employment are unlikely to find a full-time job in the current period.

Consider two competing risks where interest lies in durations to event 1 and event 2 is a competing event. Typically, we focus on the survivor function, which here is $\Pr(T > t \text{ and event 1 occurs})$. With competing risks, one instead uses the failure function, $\Pr(T \leq t \text{ and event 1 occurs})$, which is called the cumulative incidence function (CIF). For example, we consider the probability of obtaining a full-time job within t periods.

An estimate of the CIF can be obtained as follows. The appropriate hazard function includes in the risk set not only observations for which event 1 has not occurred but also those for which event 2 has already occurred. Let the random variables denoting the durations to occurrence of events 1 and 2 be denoted T_1 and T_2 . Then rather than (21.1), we consider the subhazard

$$h_1(t) = \lim_{h \rightarrow \infty} \frac{\Pr(t \leq T_1 < t + h | \text{either } T_1 > t \text{ or } T_2 \leq t)}{h}$$

Then $\text{CIF}(t) = 1 - \exp\{-H_1(t)\}$, where $H_1(t) = \int_0^t h_1(t)dt$ is the cumulative subhazard.

To motivate the expression for $\text{CIF}(t)$, note that in the absence of competing risks, we have the following relationships: the cumulative hazard function $H(t) = \int_0^t h(t)dt$, which is related to the survivor function by $H(t) = -\ln S(t)$. It follows that $S(t) = \exp\{-H(t)\}$ and hence the failure function $1 - S(t) = 1 - \exp\{-H(t)\}$.

A semiparametric competing risks version of the Cox PH model defines the subhazard function to be

$$h_1(t_i | \mathbf{x}_i) = h_{1,0}(t_i) \times \exp(\mathbf{x}'_i \boldsymbol{\beta})$$

where $h_{1,0}(t)$ is the baseline subhazard.

As before, we are interested in duration to a full-time job in which case `censor1 = 1` (for 1,073 cases). We consider the competing risk that the person obtains a job that is part-time (`censor2 = 1` in 339 cases) or obtains a job but subsequently leaves it, and we do not know whether it is part-time or full-time (`censor3 = 1` in 549 cases). The remaining persons were still unemployed (`censor4 = 1` in 1,255 cases).

The PH competing risks model can be fit using the `stcrreg` command. We obtain

```
. * Semiparametric competing risks example - full-time job versus not full-time
. qui generate event = 1 if censor1 == 1
. qui replace event = 2 if (censor2 == 1 | censor3 == 1)
. qui stset spell, fail(event=1)

. stcrreg ui logwage, compete(event == 2) nolog vce(robust)
      Failure _d: event==1
      Analysis time _t: spell

Competing-risks regression
Failure event: event == 1
Competing event: event == 2
Log pseudolikelihood = -8155.038
No. of obs      =      3,343
No. of subjects =      3,343
No. failed      =      1,073
No. competing   =       913
No. censored    =      1,357
Wald chi2(2)    =      164.45
Prob > chi2     =      0.0000
```

<code>_t</code>	Robust				
	<code>SHR</code>	<code>std. err.</code>	<code>z</code>	<code>P> z </code>	[95% conf. interval]
<code>ui</code>	.5098238	.0305227	-11.25	0.000	.4533769 .5732986
<code>logwage</code>	1.552715	.0796558	8.58	0.000	1.404184 1.716956

The subhazard-ratio coefficients in column `SHR` are interpreted similarly to the hazard ratios for the Cox PH model. In particular, because the `SHR` for `ui`

is much less than one, the hazard of finding a full-time job is much lower for persons receiving unemployment insurance, after controlling for `logwage` and the competing risk of finding a job that is not full-time.

The postestimation command `stcurve, cif` plots the cumulative incidence function. To obtain the CIF curve for the raw data, one can first estimate using the `stcrreg` command with the only regressor an intercept.

21.5 Fully parametric regression models

Fully parametric models have two advantages over the Cox PH model. First, they enable estimation of the conditional mean length of an unemployment spell and how it changes as regressors change. Second, they can provide more precise estimates of the hazard function and for both censored and uncensored observations, though different models place different restrictions on the possible shape of the hazard function.

At the same time, fully parametric models do require correct specification of the conditional density in order for parameter estimates to be consistent (the one exception being the exponential model if there is no censoring).

In the following examples, we use option `vce(robust)` to obtain heteroskedastic–robust standard errors for comparison with those from the Cox PH model. Note, however, that consistent parameter estimation in all fully parametric models, aside from the exponential with uncensored data, requires that all aspects of the distribution be correctly specified.

21.5.1 The `streg` command

The syntax for the `streg` command is

```
streg [indepvars] [if] [in] [, options]
```

Note that `indepvars` is just a list of the regressors. It is not necessary to provide the names of the dependent variable and any censoring variable because these are already declared in the `stset` command.

The censoring is assumed to be independent, meaning that after one conditions on any regressors, the censoring mechanism is independent of the duration process for the dependent variable t . The probability of being censored at time t is simply $\Pr(T > t)$.

The main option is the `distribution()` option (abbreviated `dist()`), which allows the following parametric models: exponential, generalized

gamma, Gompertz, inverse-Gaussian, loglogistic, lognormal, and Weibull distributions.

The `time` option, for the exponential and Weibull models, allows interpretation of the model as an accelerated failure-time (AFT) model rather than as a PH model.

The main postestimation commands `predict`, `margins`, and `stcurve` are illustrated in this section.

In the simplest case of single-spell single-record data, the log likelihood is similar to that for the tobit model, except that censoring is from above. Let t_i be the spell length, either completed or not completed, the binary indicator $d_i = 1$ if the spell is completed, and $f(t_i)$ and $F(t_i)$ denote the uncensored probability distribution function and c.d.f. Then the density for the i th observation is $f(t_i)^{d_i} \{1 - F(t_i)\}^{1-d_i}$ leading to log likelihood

$$\ln L = \sum_{i=1}^N [d_i \ln f(t_i) + (1 - d_i) \ln \{1 - F(t_i)\}]$$

21.5.2 Parametric Weibull model

A commonly used parametric model is the Weibull distribution, with density

$$f(t) = \gamma \alpha t^{\alpha-1} \exp(-\gamma t^\alpha) \quad (21.4)$$

The mean $E(t) = \gamma^{-1/\alpha} \Gamma(\alpha^{-1} + 1)$, where $\Gamma(\cdot)$ is the gamma function.

The corresponding hazard function is

$$h(t) = \gamma \alpha t^{\alpha-1}$$

It follows that if $\alpha > 1$, then the hazard is increasing in t ; if $\alpha < 1$, then the hazard is decreasing; and if $\alpha = 1$, then the hazard is constant. When $\alpha = 1$, the Weibull distribution reduces to the exponential distribution. As seen below, the model is too restrictive for these data because it restricts the hazard function to be monotonic.

A regression model is obtained by specifying $\gamma = \exp(\mathbf{x}'\boldsymbol{\beta})$. Then the Weibull model is a special case of the Cox PH model that defines the baseline hazard function to have the specific functional form $h_0(t) = \alpha t^{\alpha-1}$.

The Weibull model with independent censoring can be fit with the `dist(weibull)` option of the `streg` command. We obtain

```

. * Parametric Weibull regression
. stset spell, fail(censor1=1)
Survival-time data settings
    Failure event: censor1==1
Observed time interval: (0, spell]
    Exit on or before: failure



---


    3,343 total observations
        0 exclusions



---


    3,343 observations remaining, representing
    1,073 failures in single-record/single-failure data
    20,887 total analysis time at risk and under observation
                    At risk from t =          0
                    Earliest observed entry t =      0
                    Last observed exit t =       28

. streg ui logwage, nohr vce(robust) dist(weibull) nolog
    Failure _d: censor1==1
    Analysis time _t: spell
Weibull PH regression
No. of subjects = 3,343                               Number of obs = 3,343
No. of failures = 1,073
Time at risk     = 20,887                               Wald chi2(2) = 281.34
Log pseudolikelihood = -2842.8973                   Prob > chi2 = 0.0000

```

<u>_t</u>	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
ui	-1.14338	.0688903	-16.60	0.000	-1.278402 -1.008357
logwage	.4107721	.0587269	6.99	0.000	.2956695 .5258746
_cons	-4.790933	.3361208	-14.25	0.000	-5.449718 -4.132149
/ln_p	.0613705	.0182759	3.36	0.001	.0255504 .0971906
p	1.063293	.0194326			1.02588 1.10207
1/p	.9404747	.017188			.907383 .9747733

The reported parameter p equals α , so if $p > 1$, the hazard is increasing. A test that $\alpha = 1$ is the same as a test that $\ln \alpha = 0$, and from the output, this test has $z = 3.36$ and $p = 0.001$, so we strongly reject the hypothesis of a constant hazard.

From output not reported, the default standard errors of ui , $logwage$, $_cons$, and p are, respectively, 0.0631, 0.0531, 0.3102, and 0.0247, roughly

10% less than the heteroskedastic-robust standard errors.

21.5.3 Prediction and MEs

Unlike the Cox PH estimator, most parametric models can predict median and mean survival times.

The default `median time` option of the postestimation `predict` command predicts the median survival time; the `mean time` option predicts the mean survival time; and the `median lntime` and `mean lntime` options predict the natural logarithm of these quantities. The `hr` and `xb` options predict the relative hazard and $\mathbf{x}'\boldsymbol{\beta}$.

The postestimation `margins` command can use these preceding predictions, so MES for these quantities can be computed.

Additional options of the postestimation `predict` command compute the hazard (option `hazard`), survival probability (option `surv`), and various residuals: martingale-like (`mgale`), Cox–Snell (`csnell`), and deviance (`deviance`); see [ST] **streg postestimation**.

For the Weibull model, we predict the expected length of the jobless spell for each person in the sample, using the `mean time` option of the `predict` command.

```
. * Parametric Weibull model: Prediction of the conditional mean duration time  
. qui streg i.ui logwage, nohr vce(robust) dist(weibull) nolog  
. predict muspell, mean time  
. qui generate completedspell = spell if censor1==1  
. qui generate mucompletedspell = muspell if censor1==1  
. summarize muspell completedspell mucompletedspell
```

Variable	Obs	Mean	Std. dev.	Min	Max
muspell	3,343	20.34527	9.926651	4.690547	91.0024
completeds~1	1,073	4.941286	4.890527	1	27
mucomplete~1	1,073	18.14698	9.774751	4.690547	44.60316

The average of these predictions is 20.35 periods, much higher than the average across complete and incomplete spells of 6.25 periods. There is

considerable variation across individuals in the predicted spell lengths, from 4.69 periods to 91.00 periods.

We then compute the average marginal effects (AME) for the conditional mean.

```
. * Parametric Weibull model: AMEs for the conditional mean duration time
. margins, dydx(*) predict(mean time)
Average marginal effects                                         Number of obs = 3,343
Model VCE: Robust
Expression: Predicted mean _t, predict(mean time)
dy/dx wrt: 1.ui logwage
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.ui	19.33181	1.565395	12.35	0.000	16.26369	22.39992
logwage	-7.859799	1.275219	-6.16	0.000	-10.35918	-5.360415

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

The average across individuals of having unemployment insurance is to increase unemployment duration by 19.33 periods or 38.66 weeks compared with someone with no unemployment insurance.

21.5.4 Proportional hazards models

We fit a range of parametric models using the `streg` command. The list does not include the generalized gamma model because for these data, the estimates did not converge. We consider four models that lead to a hazard function that is of the PH form $h(t|\mathbf{x}) = h_0(t) \times \exp(\mathbf{x}'\boldsymbol{\beta})$.

```

. * Parametric models (plus Cox) with PH parameterization
. qui streg ui logwage, dist(exponential) nohr vce(robust)
. qui stcurve, hazard title("Exponential")
. estimates store Exponential
. qui streg ui logwage, dist(weibull) nohr vce(robust)
. qui stcurve, hazard title("Weibull")
. estimates store Weibull
. qui streg ui logwage, dist(gompertz) nohr vce(robust)
. qui stcurve, hazard title("Gompertz")
. estimates store Gompertz
. qui stcox ui logwage, nohr vce(robust)
. qui stcurve, hazard title("Cox")
. estimates store Cox

```

The resulting parameter estimates are

```

. * Estimates of parametric models with PH parameterization
. estimates table Exponential Weibull Gompertz Cox, eq(1) b(%11.3f) se
>     stats(ll aic bic)

```

Variable	Exponential	Weibull	Gompertz	Cox
#1				
ui	-1.113 0.066	-1.143 0.069	-1.094 0.065	-1.007 0.061
logwage	0.407 0.057	0.411 0.059	0.404 0.056	0.399 0.053
_cons	-4.653 0.323	-4.791 0.336	-4.577 0.317	
/ln_p		0.061 0.018		
/gamma			-0.013 0.006	
Statistics				
ll	-2846.262	-2842.897	-2844.103	-7847.299
aic	5698.523	5693.795	5696.206	15698.598
bic	5716.867	5718.253	5720.664	15710.827

Legend: b/se

The parameter estimates and their standard errors for variables `ui`, `logwage`, and `_cons` are comparable across the models. The auxiliary parameters in all

models, such as `ln_p` for the Weibull model, are all statistically significant. The Weibull model provides the best fit, though the exponential with the fewer parameters is preferred if Bayesian information criterion is used as the criterion.

21.5.5 Accelerated failure-time models

Not all hazard functions are of the PH form. An alternative class of models for duration data is that of AFT models.

An AFT model arises by modeling $\ln t$ rather than t . Then a regression model is

$$\ln t = \mathbf{x}'\boldsymbol{\beta} + u$$

where different distributions for u lead to different parametric models.

The model implies $t = v \times \exp(\mathbf{x}'\boldsymbol{\beta})$, where $v = e^u$. It follows that the hazard function partitions because $h(t|\mathbf{x}) = h_0(v) \times \exp(\mathbf{x}'\boldsymbol{\beta})$, where the baseline hazard $h_0(v)$ varies with the distribution of u but clearly does not depend on t . Now $v = t \times \exp(-\mathbf{x}'\boldsymbol{\beta})$, and making this substitution into the expression for $h(t|\mathbf{x})$ yields the following hazard function for dependent variable t , conditional on regressors \mathbf{x} , for an AFT model:

$$h(t|\mathbf{x}) = h_0\{t \exp(-\mathbf{x}'\boldsymbol{\beta})\} \times \exp(\mathbf{x}'\boldsymbol{\beta})$$

This is an acceleration of the baseline hazard if $\exp(-\mathbf{x}'\boldsymbol{\beta}) < 1$ and a deceleration otherwise.

Not all hazard functions are of the AFT form. The lognormal and loglogistic yield hazard functions of this form, and the exponential and Weibull yield hazards that can be expressed as both PH and, with some reparameterization, as AFT.

We present four models that lead to a hazard function that is of the AFT form; the models are defined in the manual entry for `streg`. The option `time` is added for the exponential and Weibull models so that results are given for the AFT parameterization rather than the default PH parameterization. We obtain

```
. * Parametric models with AFT parameterization
. qui streg ui logwage, dist(loglogistic) vce(robust)
. qui stcurve, hazard title("Loglogistic")
. estimates store Loglogistic
. qui streg ui logwage, dist(lognormal) vce(robust)
. qui stcurve, hazard title("Lognormal")
. estimates store Lognormal
. qui streg ui logwage, dist(exponential) nohr vce(robust) time
. estimates store ExponAFT
. qui streg ui logwage, dist(weibull) nohr vce(robust) time
. estimates store WeibullAFT
```

The resulting estimates are

```
. * Estimates of parametric models with AFT parameterization
. estimates table Loglogistic Lognormal ExponAFT WeibullAFT, eq(1) b(%11.3f)
>     se stats(ll aic bic)
```

Variable	Loglogistic	Lognormal	ExponAFT	WeibullAFT
#1				
ui	1.243 0.062	1.213 0.056	1.113 0.066	1.075 0.064
logwage	-0.409 0.056	-0.381 0.052	-0.407 0.057	-0.386 0.055
_cons	4.097 0.321	3.980 0.300	4.653 0.323	4.506 0.313
/lngamma	-0.301 0.019			
/lnsigma		0.237 0.019		
/ln_p				0.061 0.018
Statistics				
ll	-2774.893	-2726.227	-2846.262	-2842.897
aic	5557.787	5460.453	5698.523	5693.795
bic	5582.245	5484.912	5716.867	5718.253

Legend: b/se

Compared with the signs for PH models, the signs for the fitted coefficients are reversed because for the AFT models, the regressors enter via $\exp(-\mathbf{x}'\boldsymbol{\beta})$ rather than $\exp(\mathbf{x}'\boldsymbol{\beta})$. Aside from this sign reversal, the AFT and PH estimates for the exponential model are identical. For the Weibull model, the AFT estimates of $\boldsymbol{\beta}$ equal the PH estimates divided by p , for example, $-1.1434/1.0633 = 1.075$. The parameter estimates for variable `ui` differ across the models, though the standard errors are similar across the models. The auxiliary parameters in all models, such as `/ln_gamma` for the loglogistic model, are all statistically significant.

The lognormal model clearly provides the best fit, across both AFT and PH models. The loglogistic model is the next-best model.

21.5.6 Comparison of hazard functions

Figure 21.8 presents the fitted hazard functions from these parametric models, evaluated at the mean of the regressors, as well as the baseline hazard for Cox PH.

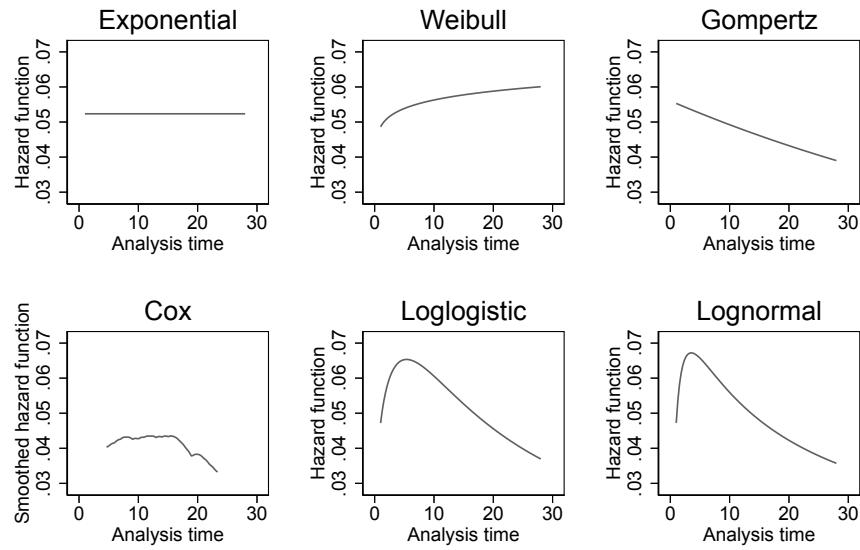


Figure 21.8. Hazard rate from various survival models

The exponential restricts the hazard function to be constant. The Weibull and Gompertz restrict the hazard function to be monotonic (increasing, decreasing, or constant). The loglogistic and lognormal allow for an inverted u-shaped hazard function, similar to that obtained earlier for the more flexible Cox PH model. As already noted, the loglogistic and lognormal provide the best fit.

21.5.7 Comparison of mean duration and associated AMEs

The conditional mean duration can be computed for four of the parametric models. We obtain

```

. * Predicted means and AMEs for four parametric models
. foreach dist in exponential weibull loglogistic lognormal {
2.     qui streg i.ui logwage, dist(`dist') vce(robust)
3.     qui margins, predict(mean time)
4.     display "Model `dist':` _col(21) "Ave mean time =" %6.2f r(b)[1,1]
5.     qui margins, dydx(*) predict(mean time)
6.     dis _col(21) "AME ui =" %6.2f r(b)[1,1] %6.2f " AME logwage= " r(b)[1,2]
7. }
Model exponential: Ave mean time = 22.10
                    AME ui = 0.00 AME logwage= 21.609415
Model weibull:      Ave mean time = 20.35
                    AME ui = 0.00 AME logwage= 19.331806
Model loglogistic: Ave mean time = 44.36
                    AME ui = 0.00 AME logwage= 47.253356
Model lognormal:    Ave mean time = 31.64
                    AME ui = 0.00 AME logwage= 33.028682

```

The difference in mean duration time is much greater than the difference in the estimates of β across the models. The biggest prediction is for the loglogistic model, which predicts an average duration across the sample of 44.36 periods, or 88.72 weeks, and one that is on average 55.15 periods higher for those with unemployment insurance.

21.5.8 Models with frailty

The parametric models require correct specification of the density for consistent estimation. One possible misspecification is that while the correct distribution is specified, the included regressors do not completely capture individual heterogeneity. Such neglected or unobserved heterogeneity, called frailty in the survival literature, may spill over to an erroneous fitted model of duration dependence.

Frailty is modeled as a multiplicative effect in the hazard function, so that the hazard function $h(t|\mathbf{x})$ becomes $\alpha \times h(t|\mathbf{x})$, where α is independent and identically distributed with either the gamma distribution or the inverse-Gaussian distribution. ML estimates are based on the density that is obtained by integrating out α .

To demonstrate the consequences of neglected heterogeneity, we introduce frailty of inverse-Gaussian form to the Weibull model, using the `frailty(invga)` option. We obtain

```

. * Weibull model with gamma frailty
. streg ui logwage, dist(weibull) frailty(invga) nolog nohr vce(robust)
      Failure _d: censor1==1
      Analysis time _t: spell
      Weibull PH regression
      Inverse-Gaussian frailty
      No. of subjects = 3,343                               Number of obs = 3,343
      No. of failures = 1,073
      Time at risk     = 20,887
      Log pseudolikelihood = -2763.0423
      Wald chi2(2) = 390.12
      Prob > chi2 = 0.0000

```

<i>_t</i>	Coefficient	Robust				
		std. err.	<i>z</i>	P> <i>z</i>	[95% conf. interval]	
ui logwage <i>_cons</i>	-1.920912	.0985825	-19.49	0.000	-2.11413	-1.727693
	.6401561	.0872424	7.34	0.000	.469164	.8111481
	-5.929082	.502098	-11.81	0.000	-6.913176	-4.944988
/ln_p /lntheta	.5138361	.0201119	25.55	0.000	.4744176	.5532546
	1.973946	.0607123	32.51	0.000	1.854952	2.09294
p 1/p theta	1.671692	.0336208			1.607078	1.738903
	.5981964	.0120308			.5750751	.6222474
	7.199027	.4370692			6.391391	8.108717

The model fit is much better. The log likelihood has increased from $-2,842.9$ to $-2,763.0$ and is now higher than for all models other than the loglogistic model.

To visualize the effect of neglected frailty on the fitted hazard rate, we compare the fitted hazard for an individual (the `alpha1` option of the `stcurve` command) with that obtained after integrating out the frailty parameter (the `unconditional` option).

```

. * Hazard curves conditional and unconditional on alpha
. qui stcurve, hazard alpha1
>      title("Conditional on {&alpha;}=1")
. qui stcurve, hazard unconditional
>      title("Unconditional on {&alpha;}")

```

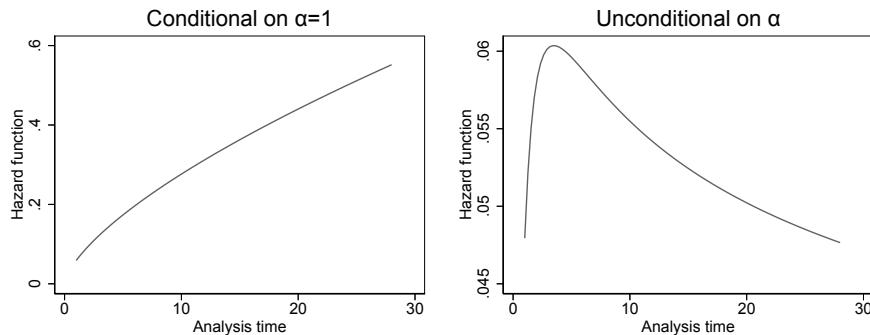


Figure 21.9. Weibull hazard controlling for frailty

Figure 21.9 presents the graph. After we control for frailty (neglected heterogeneity), the hazard rate now has an inverted u-shape similar to that obtained by the loglogistic, lognormal, and Cox PH models.

The estimated coefficients and associated standard errors are generally larger for the frailty model than those for the basic Weibull model. At the same time, there is relatively little difference in the AME at the median for the two models.

```
. * AME at median for Weibull without and with frailty
. qui streg 1.ui logwage, dist(weibull) vce(robust)
. margins, dydx(*)
note: option continuous implied because a factor with only one level was specified
      in option dydx().
Average marginal effects                                         Number of obs = 3,343
Model VCE: Robust
Expression: Predicted median _t, predict()
dy/dx wrt: 1.ui logwage
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.ui	15.87554	1.361097	11.66	0.000	13.20784	18.54324
logwage	-5.703467	.9115946	-6.26	0.000	-7.490159	-3.916774

```
. qui streg 1.ui logwage, dist(weibull) frailty(invga) vce(robust)
```

```

. margins, dydx(*)
note: option continuous implied because a factor with only one level was specified
      in option dydx().
Average marginal effects                                         Number of obs = 3,343
Model VCE: Robust
Expression: Predicted median _t, predict()
dy/dx wrt: 1.ui logwage

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.ui	16.81911	1.299592	12.94	0.000	14.27195	19.36626
logwage	-5.605074	.8508579	-6.59	0.000	-7.272725	-3.937423

21.6 Multiple-records data

The dataset studied so far is of single-record form. Datasets of multiple-record form are necessary if some regressors vary over the length of the spell or if one wishes to fit a discrete-time hazards model.

In this section, we show how to expand a single-record dataset to a multiple-record dataset and demonstrate that this data format yields the same estimates as the Cox PH model.

21.6.1 Expanding the dataset

We wish to expand the dataset with only one observation per individual to one of panel form with data for each period that the individual's spell is active. This can be done using the `expand` command, but it is simpler to use the `stssplit` command with the option `every(1)`.

We obtain

```
. * Expand data to have # observations per person = spell length  
. qui use mus221mccall, clear  
. generate id = _n // Need an individual identifier
```

```

. stset spell, fail(censor1=1) id(id) // stset must include id() option
Survival-time data settings
    ID variable: id
    Failure event: censor1==1
Observed time interval: (spell[_n-1], spell]
    Exit on or before: failure



---


    3,343  total observations
        0  exclusions


---


    3,343  observations remaining, representing
    3,343  subjects
    1,073  failures in single-failure-per-subject data
    20,887  total analysis time at risk and under observation
                    At risk from t =          0
                    Earliest observed entry t =   0
                    Last observed exit t =     28
.
sts(split t, every(1)
(17,544 observations (episodes) created)

```

Recall that there were 3,343 spells with an average length of 6.248 periods leading to a total of $3343 \times 6.248 = 20887$ periods at risk. The `sts(split` command created 17,544 additional observations for a total of $3343 + 17544 = 20887$ observations at risk.

In addition to expanding the dataset, the `sts(split` command creates several new variables and changes some existing variables. Focusing on the key variables of interest for this analysis, we have

```
. * Data created by stssplit
. summarize id spell censor1 ui logwage *_ t
```

Variable	Obs	Mean	Std. dev.	Min	Max
id	20,887	1594.551	964.6319	1	3343
spell	20,887	6.14296	5.184553	1	28
censor1	3,343	.3209692	.4669188	0	1
ui	20,887	.7062766	.4554776	0	1
logwage	20,887	5.712025	.5376671	2.70805	7.600402
_st	20,887	1	0	1	1
_d	20,887	.0513717	.2207599	0	1
_t	20,887	6.14296	5.184553	1	28
_t0	20,887	5.14296	5.184553	0	27
t	20,887	5.14296	5.184553	0	27

```
. list id spell censor1 ui logwage *_ t if id==9, clean
```

	id	spell	censor1	ui	logwage	_st	_d	_t	_t0	t
67.	9	1	.	1	5.28827	1	0	1	0	0
68.	9	2	.	1	5.28827	1	0	2	1	1
69.	9	3	.	1	5.28827	1	0	3	2	2
70.	9	4	.	1	5.28827	1	0	4	3	3
71.	9	5	.	1	5.28827	1	0	5	4	4
72.	9	6	.	1	5.28827	1	0	6	5	5
73.	9	7	1	1	5.28827	1	1	7	6	6

```
. list id spell censor1 ui logwage *_ t if id==10, clean
```

	id	spell	censor1	ui	logwage	_st	_d	_t	_t0	t
74.	10	1	.	1	5.28827	1	0	1	0	0
75.	10	2	.	1	5.28827	1	0	2	1	1
76.	10	3	.	1	5.28827	1	0	3	2	2
77.	10	4	.	1	5.28827	1	0	4	3	3
78.	10	5	0	1	5.28827	1	0	5	4	4

The spell length variable `spell` now takes value 1, 2, ... to the observed spell length. The censoring variable `censor1` is set to missing until the last time period in the spell, at which point it takes its original value of 1 if the spell actually ended with employment and 0 if the spell continued. The observations for the regressors `ui` and `logwage` are simply duplicated, so the regressors are time invariant. The new variables created include a time variable `t` equal to 0, 1, 2,

The related `stjoin` command constructs a single-record dataset from a multiple-record dataset.

Estimation with multiple-record data

The `stcox` and `streg` commands with this multiple-record data with time-invariant regressors yield the same results as those obtained earlier using the single-spell data, though computation time will be longer. For example, the `stcox` command yields

```
. * Cox PH with multiple-records data gives same results as single-record data
. stcox ui logwage, nolog vce(robust) nohr
      Failure _d: censor1==1
      Analysis time _t: spell
      ID variable: id
Cox regression with Breslow method for ties
No. of subjects = 3,343                               Number of obs = 20,887
No. of failures = 1,073
Time at risk    = 20,887
Log pseudolikelihood = -7847.2989
Wald chi2(2)   = 287.10
Prob > chi2    = 0.0000
(Std. err. adjusted for 3,343 clusters in id)
```

<code>_t</code>	Robust					
	Coefficient	std. err.	<code>z</code>	<code>P> z </code>	[95% conf. interval]	
ui	-1.007489	.0609082	-16.54	0.000	-1.126866	-.8881108
logwage	.3991956	.0525997	7.59	0.000	.296102	.5022892

The coefficient estimates, their standard errors, and the fitted log likelihood are identical to those obtained previously using the single-record data. Because there are now multiple observations per individual, the reported robust standard errors are cluster-robust with clustering on `id`.

21.7 Discrete-time hazards logit model

An alternative method for analyzing duration data is the discrete-time hazards model, which fits a binary outcome model for transitions.

Suppose we observe an individual i for eight periods, with no transition occurring in the first seven periods and in the eighth period either a transition occurs (a noncensored observation) or a transition does not occur (a censored observation). Then we have a binary dependent variable y_{it} , $t = 1, \dots, 8$, that equals 0 in the first seven periods and equals either 1 (transition) or 0 (no transition) in the final period.

The multiple-record dataset is already in suitable form, though we need to create the dependent variable for the discrete time analysis. This variable takes value 1 if a transition occurred and 0 if it did not.

```
. * Discrete-time hazards: Create indicator variable for getting employed  
. generate demploy = 0  
. replace demploy = 1 if censor1 == 1  
(1,073 real changes made)
```

We expect the variable to equal 1 for 1,073 observations because the `stset` command output stated that there were 1,073 failures.

We now fit a logit model with regressors `ui` and `logwage` as before. As already stated, in the current example, these regressors are time invariant, but more generally, the regressors can be time varying. Discrete-time hazards models additionally include a time trend. Here we include a quadratic in time.

Because multiple observations have been created per individual, valid statistical inference requires the use of cluster-robust standard errors, with clustering on the individual.

We obtain

```

. * Discrete-time hazards: logit with quadratic time trend
. logit demploy ui logwage c.t##c.t, vce(cluster id) nolog
Logistic regression                                         Number of obs = 20,887
                                                               Wald chi2(4)   = 331.19
                                                               Prob > chi2    = 0.0000
Log pseudolikelihood = -4027.6351                         Pseudo R2     = 0.0479
                                                               (Std. err. adjusted for 3,343 clusters in id)

```

demploy	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
ui	-1.100934	.0669661	-16.44	0.000	-1.232185	-.969683
logwage	.4401167	.0586368	7.51	0.000	.3251907	.5550427
t	-.1067251	.0163839	-6.51	0.000	-.138837	-.0746133
c.t#c.t	.0041279	.0008648	4.77	0.000	.002433	.0058229
_cons	-4.491509	.3323894	-13.51	0.000	-5.14298	-3.840037

The results are qualitatively similar to those from the `stcox` command with the `nohr` option. The probability of obtaining a job increases with the wage in the previous job and decreases with unemployment insurance status. This probability is initially decreasing in time before increasing after 13 periods ($\simeq -0.10673/(-2 \times 0.004128)$). The `margins` command can be used to compute MES.

21.7.1 Relationship to Cox PH model

The Cox PH model can be shown to be equivalent to a discrete-time hazards complementary log–log model where dummy variables for each time period are included as additional regressors; see, for example, [Cameron and Trivedi \(2005, chap. 17.10\)](#).

We obtain

```
. * Discrete-time hazards: cloglog with time dummies
. cloglog demploy ui logwage i.t, vce(cluster id) nolog
note: 22.t != 0 predicts failure perfectly;
      22.t omitted and 69 obs not used.

note: 23.t != 0 predicts failure perfectly;
      23.t omitted and 60 obs not used.

note: 24.t != 0 predicts failure perfectly;
      24.t omitted and 58 obs not used.

note: 27.t != 0 predicts failure perfectly;
      27.t omitted and 4 obs not used.
```

Complementary log-log regression
Number of obs = 20,696
Zero outcomes = 19,623
Nonzero outcomes = 1,073
Wald chi2(25) = 438.48
Prob > chi2 = 0.0000
Log pseudolikelihood = -3936.0639
(Std. err. adjusted for 3,343 clusters in id)

demploy	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
ui	-1.050797	.0636478	-16.51	0.000	-1.175545	-.92605
logwage	.4191098	.0552535	7.59	0.000	.3108149	.5274046
t						
1	-.2674352	.094283	-2.84	0.005	-.4522264	-.082644
2	-.4300344	.1073308	-4.01	0.000	-.6403989	-.21967
3	-.9467419	.1445455	-6.55	0.000	-1.230046	-.663438
4	-.166659	.1125021	-1.48	0.139	-.3871592	.0538411
5	-1.118788	.1852016	-6.04	0.000	-1.481777	-.7558001
6	.0064278	.1220951	0.05	0.958	-.2328743	.2457299
7	-1.491213	.2643466	-5.64	0.000	-2.009323	-.9731037
8	-.6015566	.1831592	-3.28	0.001	-.9605421	-.2425712
9	-2.854551	.5804503	-4.92	0.000	-3.992213	-1.71689
10	-.590193	.2043875	-2.89	0.004	-.9907852	-.1896008
11	-1.739855	.3828346	-4.54	0.000	-2.490197	-.9895133
12	-.3692626	.208614	-1.77	0.077	-.7781385	.0396134
13	.0298682	.1922302	0.16	0.877	-.346896	.4066324
14	-.1760333	.2389794	-0.74	0.461	-.6444243	.2923578
15	-.6208792	.3214599	-1.93	0.053	-1.250929	.0091705
16	-.5828672	.3576104	-1.63	0.103	-1.283771	.1180363
17	-.5503648	.3822622	-1.44	0.150	-1.299585	.1988553
18	-.9997678	.5044911	-1.98	0.048	-1.988552	-.0109835
19	-1.151688	.5843593	-1.97	0.049	-2.297011	-.0063648
20	-.6933875	.5080862	-1.36	0.172	-1.689218	.3024431
21	-.3726412	.5100573	-0.73	0.465	-1.372335	.6270527
22	0	(empty)				
23	0	(empty)				
24	0	(empty)				
25	-.5521873	.7122656	-0.78	0.438	-1.948202	.8438277
26	.8110057	.468242	1.73	0.083	-.1067317	1.728743
27	0	(empty)				
_cons	-4.30813	.3167992	-13.60	0.000	-4.929045	-3.687215

The omitted dummies arise because there were no transitions in some periods and to avoid the dummy variable trap.

The coefficient estimates of -1.051 and 0.419 are quite close to -1.007 and 0.399 for the Cox PH model. The corresponding standard errors are 0.064 and 0.055 compared with 0.061 and 0.053 . The coefficient estimates are also similar to those for the preceding logit model, which had a quadratic in time.

It is more common to use the logit model than the complementary log-log model. The coefficient estimates of the two models are not directly comparable but are very similar in this example. From output not listed, logit estimation of the same model yielded coefficient estimates of -1.089 and 0.439 with corresponding standard errors of 0.067 and 0.059 . Furthermore, the estimated AMEs of the two binary outcome models are very close.

```
. * Compare AMEs for discrete-time hazards cloglog and logit
. qui cloglog demploy 1.ui logwage i.t, vce(cluster id) nolog
. margins, dydx(1.ui logwage)
note: option continuous implied because a factor with only one level was specified
      in option dydx().
Average marginal effects                                         Number of obs = 20,696
Model VCE: Robust
Expression: Pr(demploy), predict()
dy/dx wrt: 1.ui logwage
```

	Delta-method				
	dy/dx	std. err.	z	P> z	[95% conf. interval]
1.ui	-.0522352	.0036189	-14.43	0.000	-.059328 -.0451423
logwage	.020834	.0028167	7.40	0.000	.0153133 .0263546

```
. qui logit demploy 1.ui logwage i.t, vce(cluster id) nolog
. margins, dydx(1.ui logwage)
note: option continuous implied because a factor with only one level was specified
      in option dydx().
Average marginal effects                                         Number of obs = 20,696
Model VCE: Robust
Expression: Pr(demploy), predict()
dy/dx wrt: 1.ui logwage
```

	Delta-method				
	dy/dx	std. err.	z	P> z	[95% conf. interval]
1.ui	-.0518829	.0035691	-14.54	0.000	-.0588781 -.0448876
logwage	.0209271	.0028518	7.34	0.000	.0153376 .0265166

We conclude that for these data, the Cox PH model, the complementary log–log model, and the logit discrete-time hazards model yield similar results.

21.8 Time-varying regressors

The `stcox` and `streg` commands allow estimation of models with regressors whose values change during the course of the spell. In that case, multiple-records data need to be used.

As an example of a single-spell model with time-varying regressors, we adjust the current multiple-records data by creating the variable `tvlogwage`, which equals the original time-invariant variable `logwage`, and performing a random draw each time period from the $N(0, 0.5^2)$ distribution.

Estimation of the Cox PH model with the time-varying regressor `tvlogwage` rather than `logwage` yields

```
. * Cox PH regression with time-varying regressors
. generate tvlogwage = logwage + 0.5*rnormal(0,1) // Create time-varying x
. stcox ui tvlogwage, vce(robust) nohr nolog
      Failure _d: censor1==1
      Analysis time _t: spell
      ID variable: id
Cox regression with Breslow method for ties
No. of subjects = 3,343                               Number of obs = 20,887
No. of failures = 1,073
Time at risk     = 20,887
Wald chi2(2)   = 259.82
Log pseudolikelihood = -7865.4094
Prob > chi2    = 0.0000
(Std. err. adjusted for 3,343 clusters in id)
```

<code>_t</code>	Robust					
	Coefficient	std. err.	<code>z</code>	<code>P> z </code>	[95% conf. interval]	
ui	-.9633531	.0599822	-16.06	0.000	-1.080916	-.8457902
tvlogwage	.1711999	.0385886	4.44	0.000	.0955677	.2468321

The coefficient of `ui` is little changed, while the coefficient of `tvlogwage` is 0.171 compared with 0.399 for `logwage`.

21.9 Clustered data

Section 13.9 presented in some detail various models and methods that can be applied when observations in the same cluster are correlated and observations in different clusters are uncorrelated. That discussion was illustrated using the Poisson model, and much of it carries over to duration data.

When there is no censoring, the simplest approach is to estimate the quasi-ML estimator of the exponential model with exponential conditional mean, using the command `glm y x, family(gamma) scale(1) link(log)` with the `vce(cluster clustvar)` option. This maintains the assumption that the conditional mean for individual i in cluster g is $E(y_{ig}|\mathbf{x}_{ig}) = \exp(\mathbf{x}'_{ig}\boldsymbol{\beta})$, the essential condition for consistency. But it provides corrected standard errors that adjust for the loss in precision that arises because of observations no longer being independent within cluster. Similarly, one could use nonlinear instrumental-variables methods for models with endogenous regressors, along with the `vce(cluster clustvar)` option. Potentially, more efficient estimates can be obtained by nonlinear feasible generalized least-squares estimation assuming equicorrelation within cluster, using the `xtgee` command with options `family(gamma), link(log), corr(exchangeable)`, and `vce(robust)`. Fixed-effects estimation leads to consistent estimates when there are many observations within each cluster. The simplest method is to add `i.clid` as regressors, where `clid` denotes the cluster identifier variable, though this can become computationally infeasible if there are many clusters.

For censored data, the Cox PH model estimator remains consistent in the presence of clustering, and one adds the `vce(cluster clustvar)` option. The `shared()` option for the `stcox` and `streg` commands introduces frailty (see section 21.5.8) at the cluster level rather than the individual level. For several parametric duration models with or without censoring, the `xtstreg` command adds cluster-specific random intercepts that are assumed to be normally distributed. The `mestreg` command generalizes this to fit multilevel MES parametric survival models for the exponential, loglogistic,

Weibull, lognormal, and gamma, with normally distributed random effects and random coefficients.

21.10 Additional resources

The [ST] *Stata Survival Analysis Reference Manual* provides a very detailed exposition of methods for survival analysis, with emphasis on methods used in biostatistics and epidemiology. The Stata Press book by [Cleves, Gould, and Marchenko \(2016\)](#) covers in much more detail many of the topics in this chapter. The website book by [Jenkins \(2005\)](#) is more oriented toward social science researchers.

Survival analysis is presented in many complete books, though relatively few are written for economists. Econometrics references include [Lancaster \(1990\)](#) and [Cameron and Trivedi \(2005, chaps. 17–19\)](#).

21.11 Exercises

1. Consider the Weibull distribution with $\gamma = 0.02$ and $\alpha = 2$. Given the formula for the Weibull density in (21.4), show that the c.d.f. is $F(t) = 1 - \exp(-0.02 \times t^2)$. Generate 10,000 uncensored Weibull observations using commands `set obs 10000` and `set seed 10101` and `t=sqrt(-ln(1-uniform())/0.02)`. Explain how the inverse transformation method is being used here. Verify that the mean of `t` is close to the theoretical mean given after (21.4). Given the formulas for $f(t)$ and $F(t)$, derive the formulas for the survivor, hazard, and integrated hazard functions for this Weibull example. Given these formulas, generate corresponding variables `ft`, `st`, `ht`, and `Ht`, and plot each of these against `t`. Now, give command `stset t`, and obtain a plot of $f(t)$ using command `kdensity t`, a plot of $S(t)$ using command `sts graph, survival ci`, and similar plots of $h(t)$ and $H(t)$. Compare these graphs to the corresponding plots of `ft`, `st`, `ht`, and `Ht` against `t`, and comment on the precision of estimation. Finally, give command `streg, dist(weibull) nohr`. Do you obtain the expected estimates?
2. Now, introduce a regressor in the example of question 1. Generate 10,000 uncensored Weibull observations with `setobs10000` and `setseed10101` and `gen x=rnormal(0,1)` and `gen t=sqrt(-ln(1-uniform())/exp(ln(0.02)+0.1*x))`. Give commands `stset t` and `streg x, dist(weibull) nohr` and `stcox x, nohr`. Do you get the expected estimates? Explain. Now, introduce random censoring using commands `gen fail=uniform()>0.6` and `stset t, failure(fail)`. Obtain Cox PH and Weibull ML estimates, and compare the estimates and precision to the uncensored case. Then, discretize the data using commands `replace t=ceil(t)` and `stset t, failure(fail)`. Obtain Cox PH and Weibull ML estimates, and compare the estimates and precision to the nondiscretized case.
3. The dataset used in this chapter includes binary variable `censor4` that equals 1 if still jobless. Give commands `use mus221mccall` and `gen fail=censor4==0` and `stset spell, failure(fail)`. What fraction of spells are completely observed? Give command `sts list`, and using the data listed under `Beg.total` and `Fail` and `Net Lost`, manually compute the estimated survivor function and hazard function at time 2.

Obtain a plot of the smoothed hazard function. Does the hazard appear to be increasing or decreasing with spell length? Consider regression with regressors `ui`, `age`, `married`, and `age`. Do the fitted coefficients obtained using the Cox PH estimator accord with your prior beliefs? Repeat using the Weibull ML estimator.

4. In the dataset of this chapter, the unemployment spell can end in several different ways. For this exercise, use the censoring variable `censor3` that equals 1 if reemployed but leaves the job. Generate survival, cumulative hazard, and hazard functions estimates and curves, with 95% confidence intervals. Compare with those using `censor1` as the censoring variable. Then, fit and tabulate the following four duration models with `ui` and `logwage` as regressors: exponential; Weibull with gamma heterogeneity (frailty); Gompertz with gamma heterogeneity (frailty); and Cox PH model. Which is the preferred model according to the `bic` criterion?
5. For the same setup as example 3, create a multiple-records dataset by expanding the dataset using `generate id=_n` and `stset spell, failure(fail) id(id)` and `stsplits t, every(1)`. Verify that applying the `stcox` command to the multiple-record dataset yields the same results as in question 3. Create a binary variable `djob` that equals 1 if the person transitions from being jobless to having a job. Estimate a discrete-time hazards logit regression of `djob` on `ui`, `age`, `married`, and a quadratic in `t`. Compare your results with those from the `stcox` command. Do the discrete-time hazards logit model estimates change much if we use dummy variables for each time period rather than a quadratic in time?

Chapter 22

Nonlinear panel models

22.1 Introduction

The general approaches to nonlinear panel models are similar to those for linear models (see chapters 8 and 9), such as pooled, population averaged (PA), random effects (RE), and fixed effects (FE).

Panel methods for nonlinear models overlap with methods for nonlinear models with clustered data. These methods have been presented in section 13.9 and in relevant sections of the various chapters for specific types of data. One difference is that for more efficient feasible generalized least-squares estimation of pooled nonlinear models with clustering, a natural model for within-cluster correlation was equicorrelation, whereas for panel data, the within correlation is likely to dampen as observations become further apart in time.

We focus on short panels. Unlike the linear case, the slope parameters in pooled and RE models differ. More generally, results for linear models do not always carry over to nonlinear models, and methods used for one type of nonlinear model may not be applicable to another type. In particular, there are only a few nonlinear models for which one can consistently estimate parameters of FE models if the panel is a short panel, and in such cases, obtaining marginal effects (MES) remains a challenge.

Overall, our coverage is more selective in the range of models considered. We begin with a general treatment of nonlinear panel models. We then give a lengthy treatment of the panel methods for binary outcome and ordered outcome models, with emphasis on the logit model. Tobit, count data, and conditional quantile regression (QR) models are then covered more concisely.

22.2 Nonlinear panel-data overview

We assume familiarity with the material in chapter 8. We use the individual-effects models as the starting point to survey the various panel methods for nonlinear models.

We consider nonlinear panel models for the scalar dependent variable y_{it} with the regressors \mathbf{x}_{it} , where i denotes the individual and t denotes time.

In some cases, a fully parametric model may be specified, with the conditional density

$$f(y_{it}|\alpha_i, \mathbf{x}_{it}) = f(y_{it}, \alpha_i + \mathbf{x}'_{it}\boldsymbol{\beta}, \gamma), \quad t = 1, \dots, T_i, \quad i = 1, \dots, N \quad (22.1)$$

where γ denotes additional model parameters such as variance parameters and α_i is an individual effect.

In other cases, a conditional mean model may be specified, with the additive effects

$$E(y_{it}|\alpha_i, \mathbf{x}_{it}) = \alpha_i + g(\mathbf{x}'_{it}\boldsymbol{\beta}) \quad (22.2)$$

or with the multiplicative effects

$$E(y_{it}|\alpha_i, \mathbf{x}_{it}) = \alpha_i \times g(\mathbf{x}'_{it}\boldsymbol{\beta}) \quad (22.3)$$

for the specified function $g(\cdot)$. In these models, \mathbf{x}_{it} includes an intercept, so α_i is a deviation from the average centered on zero in (22.1) and (22.2) and centered on unity in (22.3).

22.2.1 FE models

An FE model treats α_i as an unobserved random variable that may be correlated with the regressors \mathbf{x}_{it} . In long panels, this poses no problems, aside from possible computational challenges if there are many individuals and hence many α_i .

But in short panels, joint estimation of the fixed effects $\alpha_1, \dots, \alpha_N$ and the other model parameters, β and possibly γ , usually leads to inconsistent estimation of all parameters. The reason is that the N incidental parameters α_i cannot be consistently estimated if T_i is small, because there are only T_i observations for each α_i . This inconsistent estimation of α_i can spill over to inconsistent estimation of β .

For some models, one can eliminate α_i by appropriate conditioning on a sufficient statistic for y_{i1}, \dots, y_{iT_i} . This is the case for logit models for binary data and multinomial data and for the Poisson model for count data. For other models, aside from (22.2), it is not possible, though bias-corrected estimators have been developed; see section 22.4.8. For models of form (22.1), a simple alternative method is the correlated random-effects (CRE) model presented in section 22.2.4.

Dynamic models with individual fixed effects can be fit in some cases, most notably, conditional mean models with additive or multiplicative effects as in (22.2) and (22.3). The methods are qualitatively similar to those in the linear case. Stata does not currently provide official commands to fit dynamic nonlinear panel models; section 22.4.13 presents a community-contributed command for the logit FE model.

22.2.2 RE models

An RE model treats the individual-specific effect α_i as an unobserved random variable with the specified distribution $g(\alpha_i|\gamma)$, often the normal distribution. Then α_i is eliminated by integrating over this distribution. Specifically, the unconditional density for the i th observation is

$$f(y_{it}, \dots, y_{iT_i} | \mathbf{x}_{i1}, \dots, \mathbf{x}_{iT_i}, \beta, \gamma, \eta) = \int \left\{ \prod_{t=1}^{T_i} f(y_{it} | \mathbf{x}_{it}, \alpha_i, \beta, \gamma) \right\} g(\alpha_i | \eta) d\alpha_i \quad (22.4)$$

In nonlinear models, this integral usually has no analytical solution, but numerical integration works well because only univariate integration is required.

The RE approach can be generalized to random-slope parameters (random coefficients), not just a random intercept, with a greater computational burden because the integral is then of a higher dimension; see the `me` commands in section 23.4. RE models that allow for endogeneity and sample selection can be fit using the extended regression model commands `xteregress`, `xteprobit`, `xteoprobit`, and `xtintreg`; see section 23.7.

22.2.3 Pooled models or PA models

Pooled models set $\alpha_i = \alpha$. For parametric models, it is assumed that the marginal density for a single (i, t) pair,

$$f(y_{it}|\mathbf{x}_{it}) = f(\alpha + \mathbf{x}'_{it}\boldsymbol{\beta}, \gamma)$$

is correctly specified, regardless of the (unspecified) form of the joint density $f(y_{it}, \dots, y_{iT}|\mathbf{x}_{i1}, \dots, \mathbf{x}_{iT}, \boldsymbol{\beta}, \gamma)$. The parameter of the pooled model is easily estimated, using the cross-sectional command for the appropriate parametric model, which implicitly assumes independence over both t and i . A panel-robust or cluster-robust (with clustering on i) estimate of the variance-covariance matrix of the estimator (VCE) can then be used to correct standard errors for any dependence over time for a given individual. This approach is the analog of pooled ordinary least squares (OLS) for linear models.

Potential efficiency gains can occur if estimation accounts for the dependence over time that is inherent in panel data. This is possible for generalized linear models (GLMs), defined in section 13.3.7, where one can weight the first-order conditions for the estimator to account for correlation over time for a given individual but still have estimator consistency provided that the conditional mean is correctly specified as $E(y_{it}|\mathbf{x}_{it}) = g(\alpha + \mathbf{x}'_{it}\boldsymbol{\beta})$ for a specified function $g(\cdot)$. This is called the PA approach, or generalized estimating equations approach, and is the analogue of pooled feasible generalized least squares for linear models. For nonlinear models with clustered data, this approach is presented in section 13.9 with equicorrelation the natural model for within-cluster correlation. For panel data, it is more natural to model correlation over time for a given individual using time-series models for autocorrelation.

For linear panel-data models, the PA estimators of $\boldsymbol{\beta}$, such as OLS, and the RE estimator of $\boldsymbol{\beta}$ have the same probability limit if indeed $y_{it} = \alpha_i + \mathbf{x}'_{it}\boldsymbol{\beta} + u_{it}$ and the usual assumptions of the RE model hold. For nonlinear panel models, this generally does not hold, because the models lead to different formulas for $E(y_{it}|\mathbf{x}_{it})$. So one can no longer directly compare PA coefficient estimates with RE coefficient estimates. Instead, it is more meaningful to compare the MES, such as average marginal effects (AMES), across different models. The chapter exercise 4 demonstrates this for the probit panel model.

22.2.4 CRE models

In most nonlinear models with short panels, one cannot consistently fit an FE model because of the incidental parameters problem.

Instead, one may use the CRE model, introduced in section 8.7.4, that uses the Mundlak correction and assumes $E(\alpha_i|x_{i1}, \dots, x_{iT}) = \bar{\mathbf{x}}_i'\boldsymbol{\gamma}$ and defines $\alpha_i = \bar{\mathbf{x}}_i'\boldsymbol{\gamma} + \eta_i$, where η_i is an independent error. Then the density (22.1) becomes $f(y_{it}, \mathbf{x}_{it}'\boldsymbol{\beta} + \bar{\mathbf{x}}_i'\boldsymbol{\gamma} + \eta_i)$. The model is interpreted as a nonlinear RE model in which the RE assumptions hold conditionally on both \mathbf{x}_{it} and $\bar{\mathbf{x}}_i$. The addition of extra controls could make the RE assumption more acceptable.

This method is demonstrated for a Poisson cluster example in section 13.9.5 and for a logit panel example in section [22.4.9](#).

22.2.5 Prediction and MEs

For PA estimators of nonlinear panel models, predictions and MES are obtained in a manner similar to their calculation for the corresponding nonlinear cross-sectional model. The default for the `predict` and `margins` postestimation commands is then the conditional mean, or conditional probability in the case of a binary outcome.

Computation is more difficult, however, for FE, RE, and CRE estimators of models that introduce an individual specific effect α_i . Furthermore, the defaults for the `predict` and `margins` postestimation commands vary with the specifying nonlinear model being estimated and whether estimation is by RE or FE. These complications do not arise in linear panel models with an additive effect because, for example, if $E(y_{it}|\mathbf{x}_{it}, \alpha_i) = \mathbf{x}_{it}'\boldsymbol{\beta} + \alpha_i$, then $\partial E(y_{it}|\mathbf{x}_{it}, \alpha_i)/\partial x_{j it} = \beta_j$ does not depend on α_i

In general, $E(y_{it}|\mathbf{x}_{it}, \alpha_i) = g(\mathbf{x}_{it}, \alpha_i, \boldsymbol{\beta})$. For RE (and CRE) estimators with normally distributed random effects, the default for the `predict` postestimation command is to integrate out the random effects and predict $E(y_{it}|\mathbf{x}_{it}) = \int E(y_{it}|\mathbf{x}_{it}, \alpha_i)h(\alpha_i|\sigma^2)d\alpha_i$, where $h(\alpha_i|\sigma^2)$ is the $N(0, \sigma^2)$ density.

For those nonlinear models for which consistent FE estimation is possible in short panels, little is known about α_i , which has been eliminated by appropriate differencing. Stata commands set $\alpha_i = 0$ and predict $E(y_{it}|\mathbf{x}_{it}, \alpha_i = 0)$. Note that $E(y_{it}|\mathbf{x}_{it}, \alpha_i = 0)$ can differ greatly from $E(y_{it}|\mathbf{x}_{it})$ and $E(y_{it}|\mathbf{x}_{it}, \alpha_i)$.

MES of the conditional mean are based on the preceding predictions, $E(y_{it}|\mathbf{x}_{it})$ for RE and CRE estimators with normal random effects and $E(y_{it}|\mathbf{x}_{it}, \alpha_i = 0)$ for the FE model. For the FE model, a potentially better approach is to include individual

dummies as regressors, leading to biased parameter estimates in a short panel, and compute bias-corrected AMES; section [22.4.8](#) provides an example.

In many cases, regressors appear in the single-index form $\mathbf{x}'_{it}\beta$. Regardless of the treatment of the individual effects, if $\beta_j = 2\beta_k$, then the ME of changing the j th regressor is twice the ME of changing the k th regressor. And, as in the cross-sectional case, logit coefficients can be interpreted in terms of changes in the log-odds ratio, and Poisson coefficients can be interpreted as semielasticities with respect to the conditional mean as in the cross-sectional case.

22.2.6 Stata nonlinear panel commands

The Stata commands for PA, RE, and FE estimators of nonlinear panel models are the same as for the corresponding cross-sectional model, with the prefix `xt`. For example, `xtlogit` is the command for panel logit. The `re` option fits an RE model, the `fe` option fits an FE model if this is possible, and the `pa` option fits a PA model with correlation options detailed in section 8.4.3. The `xtgee` command with appropriate options is equivalent to the `pa` option of the `xtlogit`, `xtprobit`, and `xtpoisson` commands and is available for a wider range of models, including gamma and inverse Gaussian, for which there is no explicit `xt` command.

Models with random slopes, in addition to a random intercept, can be fit for a range of models using the mixed-effects `me` commands; see section [23.4](#). These commands are especially useful if interest lies in understanding the nature of heterogeneity across individuals.

Table [22.1](#) lists the Stata commands for pooled, PA, RE, random slopes, and FE estimators of leading nonlinear panel models. In addition to the commands listed in this table, the `xtstreg` command fits RE duration models, the `cmxtmixlogit` command fits panel mixed logit models, the `xtgee` command fits GLM models, and the `mestreg`, `meglm`, and `menl` commands enable mixed-effects estimation of, respectively, duration-data models, GLM models, and nonlinear models with additive errors.

Table 22.1. Commands for nonlinear panel models

	Binary	Multinomial	Tobit	Counts
Pooled	logit probit cloglog	ologit oprobit mlogit	tobit intreg heckman	poisson nbreg
PA	xtlogit, pa xtprobit, pa xtcloglog, pa			xtpoisson, pa xtnbreg, pa
RE	xtlogit, re xtprobit, re	xtologit, re xtoprobit, re xtmlogit, re	xttobit xtintreg, re	xtpoisson, re xtpoisson, normal xtnbreg, re
Random slopes	melogit meprobit mecloglog	meologit meoprobit	metobit meintreg	mepoisson menbreg
FE	xtlogit, fe	xtmlogit, fe		xtpoisson, fe xtnbreg, fe
Endogenous and selection	xtoprobit	xteoprobit	xteregress xtheckman xteintreg	

22.2.7 Cluster–robust inference

The `xt` commands report default standard errors that are based on correct specification of the model for any within-panel correlation. The `vce(robust)` option provides standard errors that are robust to any form of within-panel correlation, based on asymptotic theory that requires the number of individuals in the panel dataset to be large. By “within-panel correlation”, we mean correlation over time for a given individual.

The `vce(robust)` option is equivalent to `vce(cluster id)`, where `id` is the panel identifier specified in the `xtset` command. It presumes independence across individuals in the panel. Broader forms of clustering, such as individuals in households or villages, should be allowed for using the `vce(cluster clustvar)` option.

These robust standard errors provide a consistent estimate of the precision of parameter estimates. For PA estimators, consistent estimation of the model parameters requires correct specification of $E(y_{it}|\mathbf{x}_{it})$, while for most nonlinear RE estimators, the requirements for consistent estimation are more demanding and include correct specification of the distribution of within-panel correlation.

22.3 Nonlinear panel-data example

The example dataset we consider is an unbalanced panel from the Rand Health Insurance Experiment. This social experiment randomly assigned different health insurance policies to families that were followed for several years. The goal was to see how the use of health services varied with the coinsurance rate, where a coinsurance rate of 25%, for example, means that the insured pays 25% and the insurer pays 75%. Key results from the experiment were given in [Manning et al. \(1987\)](#). The data extract we use was prepared by [Deb and Trivedi \(2002\)](#).

22.3.1 Data description and summary statistics

Descriptive statistics for the dependent variables and regressors follow.

. * Describe dependent variables and regressors				
. qui use mus218rhie				
. describe dmdu med mdu lcoins ndisease female age lfam child id year				
Variable name	Storage type	Display format	Value label	Variable label
dmdu	float	%9.0g		Any MD visit; 1 if mdu>0
med	float	%9.0g		Medical exp excl outpatient men
mdu	float	%9.0g		Number face-to-face md visits
lcoins	float	%9.0g		log(coinsurance+1)
ndisease	float	%9.0g		Count of chronic diseases -- ba
female	float	%9.0g		Female
age	float	%9.0g		Age that year
lfam	float	%9.0g		Log of family size
child	float	%9.0g		Child
id	float	%9.0g		Person ID, leading digit is sit
year	float	%9.0g		Study year

The corresponding summary statistics are

```
. * Summarize dependent variables and regressors
. summarize dmdu med mdu lcoins ndisease female age lfam child id year
```

Variable	Obs	Mean	Std. dev.	Min	Max
dmdu	20,186	.6875062	.4635214	0	1
med	20,186	171.5892	698.2689	0	39182.02
mdu	20,186	2.860696	4.504765	0	77
lcoins	20,186	2.383588	2.041713	0	4.564348
ndisease	20,186	11.2445	6.741647	0	58.6
female	20,186	.5169424	.4997252	0	1
age	20,186	25.71844	16.76759	0	64.27515
lfam	20,186	1.248404	.5390681	0	2.639057
child	20,186	.4014168	.4901972	0	1
id	20,186	357971.2	180885.6	125024	632167
year	20,186	2.420044	1.217237	1	5

We consider three different dependent variables. The `dmdu` variable is a binary indicator for whether the individual visited a doctor in the current year (69% did). The `med` variable measures annual medical expenditures (in dollars), with some observations being 0 expenditures (other calculations show that 22% of the observations are 0). The `mdu` variable is the number of (face-to-face) doctor visits, with a mean of 2.9 visits. The three variables are best modeled by, respectively, logit or probit models, tobit models, and count models.

The regressors are `lcoins`, the natural logarithm of the coinsurance rate plus one; a health measure, `ndisease`; and four demographic variables. Children are included in the sample. For brevity, we do not include year dummies as regressors; their inclusion makes little difference to the estimates in this chapter.

22.3.2 Panel-data organization

For panel data, we use the `xtset` command to declare both the individual and time identifiers. By contrast, for nonpanel clustered data, one declares only the cluster identifier. The `xtdescribe` command then describes the panel-data organization.

```

. * Panel description of dataset
. xtset id year
Panel variable: id (unbalanced)
Time variable: year, 1 to 5, but with gaps
    Delta: 1 unit
. xtdescribe
    id: 125024, 125025, ..., 632167                      n =      5908
    year: 1, 2, ..., 5                                     T =       5
        Delta(year) = 1 unit
        Span(year)  = 5 periods
        (id*year uniquely identifies each observation)

Distribution of T_i:   min      5%     25%     50%     75%     95%     max
                           1        2        3        3        5        5        5
    Freq.   Percent   Cum. |   Pattern
    3710    62.80   62.80 |   111..
    1584    26.81   89.61 |   11111
    156     2.64   92.25 |   1....
    147     2.49   94.74 |   11...
    79      1.34   96.07 |   ..1..
    66      1.12   97.19 |   .11..
    33      0.56   97.75 |   ..111
    33      0.56   98.31 |   .1111
    29      0.49   98.80 |   ...11
    71      1.20  100.00 |   (other patterns)
    5908   100.00 |   XXXXX

```

The panel is unbalanced. Most individuals (90% of the sample of 5,908 individuals) were in the sample for the first three years or for the first five years, which was the sample design. There was relatively small panel attrition of about 5% over the first two years. There was also some entry, presumably because of family reconfiguration.

22.3.3 Within and between variation

Before analysis, it is useful to quantify the relative importance of within and between variation. For the dependent variables, we defer this until the relevant sections of this chapter.

The regressor variables `lcoins`, `ndisease`, and `female` are time invariant, so their within variation is zero. We therefore apply the `xtsum` command to only the other three regressors. We have

```

. * Panel summary of time-varying regressors
. xtset id year

Panel variable: id (unbalanced)
Time variable: year, 1 to 5, but with gaps
Delta: 1 unit

. xtsum age lfam child

```

Variable		Mean	Std. dev.	Min	Max	Observations
age	overall	25.71844	16.76759	0	64.27515	N = 20186
	between		16.97265	0	63.27515	n = 5908
	within		1.086687	23.46844	27.96844	T-bar = 3.41672
lfam	overall	1.248404	.5390681	0	2.639057	N = 20186
	between		.5372082	0	2.639057	n = 5908
	within		.0730824	.3242075	2.44291	T-bar = 3.41672
child	overall	.4014168	.4901972	0	1	N = 20186
	between		.4820984	0	1	n = 5908
	within		.1096116	-.3985832	1.201417	T-bar = 3.41672

For the regressors `age`, `lfam`, and `child`, most of the variation is between variation rather than within variation. We therefore expect that FE estimators will not be very efficient because they rely on within variation. Also, the FE parameter estimates may differ considerably from the other estimators if the within and between variation tell different stories.

22.3.4 FE or RE model for these data?

More generally, for these data we expect a priori that there is no need to use FE models. The point of the Rand experiment was to eliminate the endogeneity of health insurance choice, and hence endogeneity of the coinsurance rate, by randomly assigning this to individuals. The most relevant models for these data are RE or PA, which essentially just correct for the panel complication that observations are correlated over time for a given individual.

22.4 Binary outcome and ordered outcome models

We fit logit models for whether an individual visited a doctor (`dmdu`). Similar methods apply for probit and complementary log–log models using the `xtprobit` and `xtcloglog` commands, except that there is then no FE estimator.

22.4.1 Panel summary of the dependent variable

We begin by studying the correlation over time for a given individual of the binary dependent variable.

```
. * Logit: Panel summary of dependent variable  
. xtsum dmdu
```

Variable		Mean	Std. dev.	Min	Max	Observations
dmdu	overall	.6875062	.4635214	0	1	N = 20186
	between		.3571059	0	1	n = 5908
	within		.3073307	-.1124938	1.487506	T-bar = 3.41672

The dependent variable `dmdu` has within variation and between variation of similar magnitude.

The `xttrans` command summarizes transitions from one period to the next.

```
. * Year-to-year transitions in whether visit doctor  
. xttrans dmdu
```

Any MD visit; 1 if if mdu>0	Any MD visit; 1 if mdu>0		Total
	0	1	
0	58.87	41.13	100.00
1	19.73	80.27	100.00
Total	31.81	68.19	100.00

There is considerable persistence from year to year: 59% of those who did not visit a doctor one year also did not visit the next, while 80% of those who did visit a doctor one year also visited the next.

The `pwcorr` command gives the time-series correlation of the dependent variable.

	dmdu	L.dmdu	L2.dmdu	L3.dmdu	L4.dmdu
dmdu	1.0000				
L.dmdu	0.3884	1.0000			
L2.dmdu	0.3599	0.3807	1.0000		
L3.dmdu	0.3351	0.3400	0.3563	1.0000	
L4.dmdu	0.3329	0.3221	0.3372	0.3503	1.0000

The correlations in the dependent variable, `dmdu`, vary little with lag length, unlike the chapter 8 example of log wage where correlations decrease as lag length rises. Such constancy is unusual for panel data and suggests that estimators that impose equicorrelation may be quite reasonable for these data.

22.4.2 Pooled logit estimator

The pooled logit model is the usual cross-sectional model,

$$\Pr(y_{it} = 1 | \mathbf{x}_{it}) = \Lambda(\mathbf{x}'_{it} \boldsymbol{\beta}) \quad (22.5)$$

where $\Lambda(z) = e^z / (1 + e^z)$. A cluster-robust estimate for the VCE is then used to correct for error correlation over time for a given individual.

The `logit` command with the `vce(cluster id)` option yields

```
. * Logit cross-section with panel-robust standard errors
. logit dmdu lcoins ndisease female age lfam child, vce(cluster id) nolog
Logistic regression                                         Number of obs = 20,186
Log pseudolikelihood = -11973.392                         Wald chi2(6)    = 488.18
                                                               Prob > chi2    = 0.0000
                                                               Pseudo R2     = 0.0450
                                                               (Std. err. adjusted for 5,908 clusters in id)
```

dmdu	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
lcoins	-.1572107	.0109064	-14.41	0.000	-.1785869	-.1358345
ndisease	.050301	.0039657	12.68	0.000	.0425285	.0580735
female	.3091573	.0445772	6.94	0.000	.2217876	.396527
age	.0042689	.0022307	1.91	0.056	-.0001032	.008641
lfam	-.2047573	.0470287	-4.35	0.000	-.2969317	-.1125828
child	.0921709	.0728107	1.27	0.206	-.0505355	.2348773
_cons	.6039411	.1107712	5.45	0.000	.3868335	.8210486

The first four regressors have the expected signs. The negative sign of `lfam` may be due to family economies of scale in healthcare. The positive coefficient of `child` may reflect a u-shaped pattern of doctor visits with `age`. The estimates imply that a child of age 10, say, is as likely to see the doctor as a young adult of age 31 because $0.092 + 0.0043 \times 10 \simeq 0.0043 \times 31 = 0.1333$.

The estimated coefficients can be converted to MES by using the `margins`, `dydx(*)` command or, approximately, by multiplying by $\bar{y}(1 - \bar{y}) = 0.69 \times 0.31 = 0.21$. For example, the probability of a doctor visit at some stage during the year is 0.07 higher for a woman than for a man because $0.31 \times 0.21 = 0.07$.

In output not given, the default standard errors are approximately two-thirds those given here, so the use of cluster-robust standard errors is necessary.

22.4.3 The `xtlogit` command

The pooled logit command assumes independence over i and t , leading to potential efficiency loss, and ignores the possibility of fixed effects that would lead to inconsistent parameter estimates.

These panel complications are accommodated by the `xtlogit` command, which has the syntax

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

The options are for PA (`pa`), RE (`re`), and FE (`fe`) estimators. Panel-robust standard errors for PA and RE estimators can be calculated by using the `vce(robust)` option. The FE estimator requires the assumption of within-panel independence, so the `vce(robust)` option is not available. Model-specific options are discussed below in the relevant model section, and postestimation prediction and MES are presented in section [22.4.11](#).

22.4.4 The `xtgee` command

The `pa` option for the `xtlogit` command is also available for some other nonlinear panel commands, such as `xtpoisson`. It is a special case of the `xtgee` command for generalized estimating equations, a cluster generalization of GLMs. This command has the syntax

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

The `family()` and `link()` options define the specific model. For example, the linear model is `family(gaussian) link(identity)`; the logit model is `family(binomial) link(logit)`. Other `family()` options are `poisson`, `nbinomial`, `gamma`, and `igaussian` (inverse Gaussian).

The `corr()` option defines the pattern of time-series correlation assumed for observations on the i th individual. These patterns include `exchangeable` for equicorrelation, `independent` for no correlation, and various time-series models that have been detailed in section 8.4.3.

In the examples below, we obtain the PA estimator by using commands such as `xtlogit` with the `pa` option. If instead the corresponding `xtgee` command is used, then the `estat wcorrelation` postestimation command produces the estimated matrix of the within-group correlations.

22.4.5 PA logit estimator

The PA estimator of the parameters of (22.5) can be obtained by using the `xtlogit` command with the `pa` option. Different arguments for the `corr()` option, presented in section 8.4.3 and in [XT] `xtgee`, correspond to different models for the correlation

$$\rho_{ts} = \text{Cor}[\{y_{it} - \Lambda(\mathbf{x}'_{it}\boldsymbol{\beta})\}\{y_{is} - \Lambda(\mathbf{x}'_{is}\boldsymbol{\beta})\}], \quad s \neq t$$

The `corr(independent)` option gives the pooled logit estimator.

The exchangeable model assumes that correlations are the same regardless of how many years apart the observations are, so $\rho_{ts} = \rho$. For our data, this model may be adequate because, from section 22.4.1, the correlations of `dmdu` varied little with the lag length. Even with equicorrelation, the covariances can vary across individuals and across year pairs because, given $\text{Var}(y_{it}|\mathbf{x}_{it}) = \Lambda_{it}(1 - \Lambda_{it})$, the implied covariance is $\rho\sqrt{\Lambda_{it}(1 - \Lambda_{it})} \times \Lambda_{is}(1 - \Lambda_{is})$.

Estimation with the `xtlogit, pa corr(exch)` command yields

```

. * Pooled logit cross-section with exchangeable errors and panel-robust VCE
. xtlogit dmdu lcoins ndisease female age lfam child, pa corr(exch)
> vce(robust) nolog

GEE population-averaged model
Number of obs      = 20,186
Group variable: id
Number of groups   = 5,908
Family: Binomial
Obs per group:
Link: Logit
min =      1
Correlation: exchangeable
avg =      3.4
max =      5
Wald chi2(6)     = 521.45
Scale parameter = 1
Prob > chi2      = 0.0000
(Std. err. adjusted for clustering on id)

```

dmdu	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
lcoins	-.1603179	.0107779	-14.87	0.000	-.1814422	-.1391935
ndisease	.0515445	.0038528	13.38	0.000	.0439931	.0590958
female	.2977003	.0438316	6.79	0.000	.211792	.3836086
age	.0045675	.0021001	2.17	0.030	.0004514	.0086836
lfam	-.2044045	.0455004	-4.49	0.000	-.2935837	-.1152254
child	.1184697	.0674367	1.76	0.079	-.0137039	.2506432
_cons	.5776986	.106591	5.42	0.000	.368784	.7866132

The pooled logit and PA logit parameter estimates are very similar. The cluster-robust standard errors are slightly lower for the PA estimates, indicating a slight efficiency gain. The parameter estimates can be interpreted in exactly the same way as those from a cross-sectional logit model.

The within correlations are stored in the $e(R)$ matrix. We have

```

. * Within correlation of the exchangeable errors
. matrix list e(R)

symmetric e(R)[5,5]
      c1          c2          c3          c4          c5
r1    1
r2  .34033336        1
r3  .34033336  .34033336        1
r4  .34033336  .34033336  .34033336        1
r5  .34033336  .34033336  .34033336  .34033336        1

```

So $\hat{\rho}_{ts} = \hat{\rho} = 0.34$.

22.4.6 RE logit estimator

The logit individual-effects model specifies that

$$\Pr(y_{it} = 1 | \mathbf{x}_{it}, \boldsymbol{\beta}, \alpha_i) = \Lambda(\alpha_i + \mathbf{x}'_{it}\boldsymbol{\beta}) \quad (22.6)$$

where α_i may be a fixed effect or a random effect.

The logit RE model specifies that $\alpha_i \sim N(0, \sigma_\alpha^2)$. Then the joint density for the i th observation, after integrating out α_i , is

$$f(y_{it}, \dots, y_{iT}) = \int \left[\prod_{t=1}^{T_i} \Lambda(\alpha_i + \mathbf{x}'_{it}\boldsymbol{\beta})^{y_{it}} \{1 - \Lambda(\alpha_i + \mathbf{x}'_{it}\boldsymbol{\beta})\}^{1-y_{it}} \right] g(\alpha_i | \sigma^2) d\alpha_i \quad (22.7)$$

where $g(\alpha_i | \sigma^2)$ is the $N(0, \sigma_\alpha^2)$ density. After α_i is integrated out, $\Pr(y_{it} = 1 | \mathbf{x}_{it}, \boldsymbol{\beta}) \neq \Lambda(\mathbf{x}'_{it}\boldsymbol{\beta})$, so the RE model parameters are not comparable with those from pooled logit and PA logit.

There is no analytical solution to the univariate integral (22.7), so numerical methods are used. The default method is adaptive 12-point Gauss–Hermite quadrature. The `intmethod()` option allows other quadrature methods to be used, and the `intpoints()` option allows the use of a different number of quadrature points. The `quadchk` command checks whether a good approximation has been found by using a different number of quadrature points and comparing solutions; see [XT] **xtlogit** and [XT] **quadchk** for details.

The RE estimator is implemented by using the `xtlogit` command with the `re` option. We have

```

. * Logit RE estimator
. xtlogit dmdu lcoins ndisease female age lfam child, re nolog vce(robust)
Calculating robust standard errors ...
Random-effects logistic regression
Group variable: id
Random effects u_i ~ Gaussian
Number of obs      = 20,186
Number of groups  = 5,908
Obs per group:
    min =       1
    avg =     3.4
    max =       5
Integration method: mvaghermite
Integration pts. =      12
Wald chi2(6)      = 526.83
Log pseudolikelihood = -10878.687
Prob > chi2       = 0.0000
(Std. err. adjusted for 5,908 clusters in id)

```

dmdu	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
lcoins	-.2403864	.016318	-14.73	0.000	-.272369 -.2084038
ndisease	.078151	.0057388	13.62	0.000	.0669032 .0893988
female	.4631005	.0668151	6.93	0.000	.3321454 .5940555
age	.0073441	.0031756	2.31	0.021	.0011199 .0135682
lfam	-.3021841	.0675885	-4.47	0.000	-.4346551 -.169713
child	.1935357	.101308	1.91	0.056	-.0050243 .3920956
_cons	.8629898	.1598343	5.40	0.000	.5497203 1.176259
/lnsig2u	1.225652	.0495481			1.12854 1.322765
sigma_u	1.84564	.045724			1.758164 1.937469
rho	.5087003	.0123833			.4844281 .5329316

The RE model fits much better than the cross-sectional logit model, with log likelihood increasing from -11973.4 to -10878.7 . The coefficient estimates are roughly 50% larger in absolute value than those of the PA model. The standard errors are also roughly 50% larger, so the t statistics are little changed. Clearly, the RE model has a different conditional mean than the PA model, and the parameters are not directly comparable.

Computation and comparison of MES is presented in section [22.4.11](#). Without obtaining MES, note that from [\(22.7\)](#), $\Pr(y_{it} = 1 | \mathbf{x}_{it}, \boldsymbol{\beta}, \alpha_i)$ is of single-index form. As discussed in section [22.2.5](#), if one coefficient is twice as big as another, then the MES are twice as large.

The output also includes `sigma_u`, the estimate of the standard deviation σ_α of the random effects, so it is estimated that $\alpha_i \sim N(0, 1.846^2)$. The logit RE model can be motivated as coming from a latent-variable model, with $y_{it} = 1$ if $y_{it}^* = \mathbf{x}'_{it}\boldsymbol{\beta} + \alpha_i + \varepsilon_{it} > 0$, where ε_{it} is logistically distributed with a variance of $\sigma_\varepsilon^2 = \pi^2/3$. By a calculation similar to that in section 8.3.10, the intraclass error

correlation in the latent-variable model is $\rho = \sigma_\alpha^2 / (\sigma_\alpha^2 + \sigma_\varepsilon^2)$. Here $\hat{\rho} = 1.846^2 / (1.846^2 + \pi^2/3) = 0.509$, the quantity reported as `rho`.

22.4.7 FE logit estimator

In the FE model, the α_i may be correlated with the covariates in the model. Parameter estimation is difficult, and many of the approaches in the linear case fail. In particular, the least-squares dummy-variable (DV) estimator of section 8.5.4 yielded a consistent estimate of β , but a similar DV estimator for the logit model leads to inconsistent estimation of β in the logit model, unless $T \rightarrow \infty$.

One method of consistent estimation eliminates the α_i from the estimation equation. This method is the conditional maximum-likelihood (ML) estimator, which is based on a log density for the i th individual that conditions on $\sum_{t=1}^{T_i} y_{it}$, the total number of outcomes equal to 1 for a given individual over time.

We demonstrate this in the simplest case of two time periods. Condition on $y_{i1} + y_{i2} = 1$, so that $y_{it} = 1$ in exactly one of the two periods. Then, in general,

$$\Pr(y_{i1} = 0, y_{i2} = 1 | y_{i1} + y_{i2} = 1) = \frac{\Pr(y_{i1} = 0, y_{i2} = 1)}{\Pr(y_{i1} = 0, y_{i2} = 1) + \Pr(y_{i1} = 1, y_{i2} = 0)} \quad (22.8)$$

Now $\Pr(y_{i1} = 0, y_{i2} = 1) = \Pr(y_{i1} = 0) \times \Pr(y_{i2} = 1)$, assuming that y_{1i} and y_{2i} are independent given α_i and \mathbf{x}_{it} . For the logit model (22.6), we obtain

$$\Pr(y_{i1} = 0, y_{i2} = 1) = \frac{1}{1 + \exp(\alpha_i + \mathbf{x}'_{i1}\beta)} \times \frac{\exp(\alpha_i + \mathbf{x}'_{i2}\beta)}{1 + \exp(\alpha_i + \mathbf{x}'_{i2}\beta)}$$

Similarly,

$$\Pr(y_{i1} = 1, y_{i2} = 0) = \frac{\exp(\alpha_i + \mathbf{x}'_{i1}\beta)}{1 + \exp(\alpha_i + \mathbf{x}'_{i1}\beta)} \times \frac{1}{1 + \exp(\alpha_i + \mathbf{x}'_{i2}\beta)}$$

When we substitute these two expressions into (22.8), the denominators cancel, and we obtain

$$\begin{aligned}
\Pr(y_{i1} = 0, y_{i2} = 1 | y_{i1} + y_{i2} = 1) \\
&= \exp(\alpha_i + \mathbf{x}'_{i2}\boldsymbol{\beta}) / \{\exp(\alpha_i + \mathbf{x}'_{i1}\boldsymbol{\beta}) + \exp(\alpha_i + \mathbf{x}'_{i2}\boldsymbol{\beta})\} \\
&= \exp(\mathbf{x}'_{i2}\boldsymbol{\beta}) / \{\exp(\mathbf{x}'_{i1}\boldsymbol{\beta}) + \exp(\mathbf{x}'_{i2}\boldsymbol{\beta})\} \\
&= \exp\{(\mathbf{x}_{i2} - \mathbf{x}_{i1})'\boldsymbol{\beta}\} / [1 + \exp\{(\mathbf{x}_{i2} - \mathbf{x}_{i1})'\boldsymbol{\beta}\}]
\end{aligned} \tag{22.9}$$

There are several results. First, conditioning eliminates the problematic fixed effects α_i . Second, the resulting conditional model is a logit model with the regressor $\mathbf{x}_{i2} - \mathbf{x}_{i1}$. Third, coefficients of time-invariant regressors are not identified, because then $x_{i2} - x_{i1} = 0$.

More generally, with up to T outcomes, we can eliminate α_i by conditioning on $\sum_{t=1}^T y_{it} = 1$ and on $\sum_{t=1}^T y_{it} = 2, \dots, \sum_{t=1}^T y_{it} = T - 1$. This leads to the loss of those observations where y_{it} is 0 for all t or y_{it} is 1 for all T . The resulting conditional model is more generally a multinomial logit model. For details, see, for example, [Cameron and Trivedi \(2005, 796–797\)](#) or [R] **clogit**.

The FE estimator is obtained by using the `xtlogit` command with the `fe` option. We have

```

. * Logit FE estimator
. xtlogit dmdu lcoins ndisease female age lfam child, fe nolog
note: multiple positive outcomes within groups encountered.
note: 3,459 groups (11,161 obs) omitted because of all positive or
      all negative outcomes.
note: lcoins omitted because of no within-group variance.
note: ndisease omitted because of no within-group variance.
note: female omitted because of no within-group variance.

Conditional fixed-effects logistic regression           Number of obs     =  9,025
Group variable: id                                    Number of groups  =  2,449
                                                       Obs per group:
                                                       min =          2
                                                       avg =        3.7
                                                       max =          5
                                                       LR chi2(3)    =   10.74
Log likelihood = -3395.5996                         Prob > chi2    =  0.0132

```

dmdu	Coefficient	Std. err.	z	P> z	[95% conf. interval]
lcoins	0	(omitted)			
ndisease	0	(omitted)			
female	0	(omitted)			
age	-.0341815	.0183827	-1.86	0.063	-.070211 .001848
lfam	.478755	.2597327	1.84	0.065	-.0303116 .9878217
child	.270458	.1684974	1.61	0.108	-.0597907 .6007068

As expected, coefficients of the time-invariant regressors are not identified, and these variables are dropped. The 3,459 individuals with $\sum_{i=1}^{T_i} y_{it} = 0$ (all 0s) or $\sum_{i=1}^{T_i} y_{it} = T_i$ (all 1s) are dropped because there is then no variation in y_{it} over t , leading to a loss of 11,161 of the original 20,186 observations. Standard errors are substantially larger for FE estimation because of this loss of observations and because only within variation of the regressors is used.

The coefficients are considerably different from those for the RE logit model, and in two cases, the sign changes. The interpretation of parameters is similar to that given at the end of section [22.4.6](#) for the RE model.

22.4.8 Bias-corrected logit dummy-variable FE estimator

The logit DV estimator of the FE model obtains logit estimates of both the slope parameters β and the fixed effects $\alpha_i, i = 1, \dots, N$. In a short panel, this leads to inconsistent estimates of all parameters and subsequent MES.

[Hahn and Newey \(2004\)](#) present general methods for bias reduction using analytical formulas or using the jackknife. [Fernández-Val \(2009\)](#) specialized these methods to obtain bias-corrected estimates of both parameters and the consequent AMES for static and dynamic logit and probit models. For model parameters, the order of the bias is reduced from $O(T^{-1})$ to $O(T^{-2})$ under the assumption that $T/N^{1/3} \rightarrow \infty$. [Fernández-Val and Weidner \(2016\)](#) considered the long panel case with both time individual fixed effects and time fixed effects, where both $T \rightarrow \infty$ and $N \rightarrow \infty$ and obtained bias corrections assuming $T/N \rightarrow c$ for some constant c .

The `logitfe` and `probitfe` commands ([Cruz-Gonzalez, Fernández-Val, and Weidner 2017](#)) provide jackknife and analytical bias-corrected estimates of parameters and MES in FE logit and probit models in short and long panels.

The following command provides short-panel analytical bias-corrected estimates of parameters and AMES in the logit FE model.

```
. global xlist lcoins ndisease female age lfam child
. * Logit FE DV estimator with analytical bias correction
. logitfe ddu $xlist, teffects(no) analytical
  (output omitted)
. estimates store FEDVBC
```

The parameter estimates are given in section [22.4.10](#), and AMES are given in section [22.4.11](#). The option `nocorrection` gives estimates without the bias

correction and is equivalent to `logit dmdu i.id $xlist`. The option `jackknife` gives bias-corrected estimates obtained using the panel delete-one-jackknife.

22.4.9 CRE logit estimator

The logit CRE estimator is obtained as the logit RE estimator with inclusion as additional regressors the means over time of regressors that vary over both individual and time. We have

```
. * Logit-correlated RE estimator
. bysort id: egen aveage = mean(age)
. by id: egen avelfam = mean(lfam)
. by id: egen avechild = mean(child)
. xtlogit dmdu $xlist aveage avelfam avechild, re vce(robust)
  (output omitted)
. estimates store CRE
```

The parameter estimates are given in the next subsection.

22.4.10 Comparison of panel logit parameter estimates

We combine the preceding estimators into a single table that makes comparison easier. The models are

```
. * Panel logit estimator comparison
. global xlist lcoins ndisease female age lfam child
. qui logit dmdu $xlist, vce(cluster id)
. estimates store POOLED
. qui xtlogit dmdu $xlist, pa corr(exch) vce(robust)
. estimates store PA
. qui xtlogit dmdu $xlist, re vce(robust)
. estimates store RE
. qui xtlogit dmdu $xlist, fe          // vce(robust) not available
. estimates store FE
. qui logitfe dmdu $xlist, teffects(no) nocorrection
. estimates store FEDV
. qui logitfe dmdu $xlist, teffects(no) analytical
. estimates store FEDVBC
. qui xtlogit dmdu $xlist aveage avelfam avechild, re vce(robust)
. estimates store CRE
```

The slope parameter estimates are

```
. * Panel logit estimator comparison results
. estimates table PA RE FE FEDV FEDVBC CRE, equations(1) se b(%7.4f)
>     stats(N ll) stfmt(%7.0f) varwidth(7)
```

Variable	PA	RE	FE	FEDV	FEDVBC
#1					
lcoins	-0.1603 0.0108	-0.2404 0.0163	(omitted)	(omitted)	(omitted)
ndisease	0.0515 0.0039	0.0782 0.0057	(omitted)	(omitted)	(omitted)
female	0.2977 0.0438	0.4631 0.0668	(omitted)	(omitted)	(omitted)
age	0.0046 0.0021	0.0073 0.0032	-0.0342 0.0184	-0.0453 0.0212	-0.0351 0.0212
lfam	-0.2044 0.0455	-0.3022 0.0676	0.4788 0.2597	0.6507 0.3033	0.4809 0.3033
child	0.1185 0.0674	0.1935 0.1013	0.2705 0.1685	0.3616 0.1955	0.2749 0.1955
aveage					
avelfam					
avechild					
_cons	0.5777 0.1066	0.8630 0.1598			
/lnsig2u		1.2257 0.0495			
Statist~s					
N	20186	20186	9025	9025	9025
ll		-10879	-3396	-5515	-5515

Legend: b/se

Variable	CRE
#1	
lcoins	-0.2412 0.0163
ndisease	0.0784 0.0058
female	0.4620 0.0670
age	-0.0342 0.0186
lfam	0.5008 0.2743
child	0.2950 0.1783
aveage	0.0404 0.0190
avelfam	-0.8380 0.2834
avechild	-0.1511 0.2174
_cons	0.9578 0.1716
/lnsig2u	1.2289 0.0496
Statist~s	
N	20186
ll	-10871

Legend: b/se

The pooled logit estimates with cluster-robust standard errors are omitted from the table but are very similar to the PA estimates. The RE logit estimates differ quite substantially from the PA logit estimates, though, as already noted, the associated *t* statistics are quite similar.

The remaining columns present various estimates of the FE model. The consistent FE estimates are much less precise than the preceding estimates and are available only for time-varying regressors. Note also that panel-robust standard errors are unavailable for the logit FE estimator because its consistency requires that observations be independent over time for a given individual once the fixed effect is included. The logit DV estimates (column `FEDV`) differ substantially from the FE estimates because of incidental parameters bias, which is large here with at most five time periods. The bias-corrected DV estimates (column `FEDVBC`) are within 10% of the FE estimates with standard errors that are around 20% larger. The CRE estimates for the time-varying regressors `age`, `lfam`, and `child` are within 10% of

the FE estimates, as are the corresponding standard errors. In this example, the various FE estimators, aside from inconsistent logit FEDVs, give similar results.

22.4.11 Panel logit prediction and MEs

The `predict` postestimation command has several options that vary depending on whether the `xtlogit` command was used with the `pa`, `re`, or `fe` option. The `if` qualifier of the `predict` and `margins` commands can be used to obtain predictions and MES for specific years.

After the `xtlogit, pa` command, the default `predict` option is `mu`, which gives the predicted probability given in (22.5). After the `xtlogit, re` command, the default `predict` option is `xb`. This gives the marginal or unconditional probability that integrates out α_i , so $\Pr(y_{it} = 1 | \mathbf{x}_{it}, \boldsymbol{\beta}) = \int \Lambda(\alpha_i + \mathbf{x}'_{it}\boldsymbol{\beta})h(\alpha_i|\sigma^2)d\alpha_i$, where $h(\alpha_i|\sigma^2)$ is the $N(0, \sigma^2)$ density. After the `xtlogit, fe` command, the default `predict` option is `pc1`, which produces the conditional probability that $y_{it} = 1$ given that exactly 1 of y_{i1}, \dots, y_{iT_i} equals 1. This is used because this conditional probability does not depend on α_i ; the formula in the special case $T_i = 2$ is given in (22.9). An alternative `predict` option is `pu0`, which gives the predicted probability when $\alpha_i = 0$.

The following summarizes the default AMEs for the PA, RE, and FE logit models. For PA and RE models, these are based on the `predict` defaults, while for the FE model, the `margins` default is `pu0`, which gives the MES when $\alpha_i = 0$ (`pr1` is not available).

```
. * AMEs for PA, RE, and FE estimates
. foreach model in pa re fe {
    2.      qui xtlogit dmdu $xlist, `model'
    3.      qui margins, dydx(*)
    4.      display "Model `model':" _col(11) "lcoins" _col(21) "ndisease"
    >          _col(31) "female" _col(41) "age" _col(51) "lfam" _col(61) "child"
    5.      display _col(11) %7.4f r(b)[1,1] _col(21) %7.4f r(b)[1,2]
    >          _col(31) %7.4f r(b)[1,3] _col(41) %7.4f r(b)[1,4]
    >          _col(51) %7.4f r(b)[1,5] _col(61) %7.4f r(b)[1,6]
    6.  }
Model pa: lcoins      ndisease     female      age       lfam      child
        -0.0326      0.0105      0.0606      0.0009     -0.0416      0.0241
Model re: lcoins      ndisease     female      age       lfam      child
        -0.0320      0.0104      0.0617      0.0010     -0.0403      0.0258
Model fe: lcoins      ndisease     female      age       lfam      child
        0.0000      0.0000      0.0000     -0.0075      0.1047      0.0591
```

The AMEs following PA and RE estimation are very similar, and those for the PA estimator are much simpler to compute, as emphasized by [Drukker \(2008\)](#). The

AMES following FE estimation differ substantially because they are for $\Pr(y_{it} = 1 | \mathbf{x}_{it}, \alpha_i = 0)$ rather than for $\Pr(y_{it} = 1 | \mathbf{x}_{it})$.

For the FE model, the AMES for $\Pr(y_{it} = 1 | \mathbf{x}_{it})$ can be obtained using the `logitfe` command with bias correction. For brevity, we omit the intermediate lengthy output that includes parameter estimates. We obtain

```
. * AMEs for bias-corrected DV FE estimator
. logitfe dmdu $xlist, teffects(no) analytical
(output omitted)
```

Average Partial Effects

dmdu	Coefficient	Std. err.	z	P> z	[95% conf. interval]
lcoins	0	(omitted)			
ndisease	0	(omitted)			
female	0	(omitted)			
age	-.0055163	.004079	-1.35	0.176	-.013511 .0024784
lfam	.0756653	.0585377	1.29	0.196	-.0390665 .1903972
child	.0427447	.0180019	2.37	0.018	.0074615 .0780278

The AMES differ from the preceding AMES for the FE estimator, as expected. They also differ from those following PA and RE estimation, including two sign changes, though the sign changes are for AMES that are statistically insignificant at 5%.

A faster alternative is to fit the CRE model. We obtain

```
. * AMEs for CRE estimator
. qui xtlogit dmdu $xlist aveage avelfam avechild, re vce(robust)
. margins, dydx(*)

Average marginal effects                                         Number of obs = 20,186
Model VCE: Robust

Expression: Pr(dmdu=1), predict(pr)
dy/dx wrt: lcoins ndisease female age lfam child aveage avelfam avechild
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
lcoins	-.0320821	.0020815	-15.41	0.000	-.0361618	-.0280024
ndisease	.0104306	.0007412	14.07	0.000	.0089779	.0118834
female	.0614568	.008835	6.96	0.000	.0441404	.0787731
age	-.0045543	.0024691	-1.84	0.065	-.0093936	.0002849
lfam	.0666244	.0364848	1.83	0.068	-.0048844	.1381332
child	.0392392	.0237247	1.65	0.098	-.0072603	.0857387
aveage	.0053778	.00252	2.13	0.033	.0004387	.0103169
avelfam	-.1114862	.0376606	-2.96	0.003	-.1852996	-.0376727
avechild	-.0200974	.0289229	-0.69	0.487	-.0767852	.0365905

The AMEs for `age`, `lfam`, and `child` are, respectively, -0.0046 , 0.0666 , and 0.0392 , quite similar to -0.0055 , 0.0757 , and 0.0427 for the bias-corrected DV estimator of the FE model.

22.4.12 Mixed-effects logit estimator

The RE logit model specifies only the intercept to be normally distributed. Slope parameters may also be normally distributed. The `melogit` command estimates the parameters of this model, which is the logit extension of `mixed` for linear models, presented in section 6.7.3.

For example, the following yields the same estimates as `xtlogit` with the `re` option, aside from minor computational difference:

```
. * Following identical to xtlogit, re command
. melogit dmdu lcoins ndisease female age lfam child || id:
  (output omitted)
```

Adding the `lcoins` and `ndisease` variables after `id:` allows the intercept and slope parameters for `lcoins` and `ndisease` to be jointly normally distributed. Then a trivariate integral is computed with the Gauss–Hermite quadrature, and estimation is very computationally intensive; without restrictions on the variance matrix, the model may not even be estimable.

As in the linear case, the mixed logit model is used more for clustered data than for panel data and is mostly used in areas of applied statistics other than econometrics.

22.4.13 Dynamic panel logit models

Many current discrete economic outcomes are partly determined by past outcomes. Reasons underlying such dependence include the force of inertia, costs of adjustment, habit persistence, and true state dependence. For example, the probability that a currently unemployed person would remain unemployed in the next period may depend on previous unemployment duration.

There are several ways of modeling dynamic dependence, two of which are standard. The simplest and most direct is to capture dependence on past values of exogenous variables by including them as regressors; for example, the function $\Pr(y_{it} = 1|x_{it}, x_{it-1})$ implies that a change in x has both a contemporaneous and a lagged effect. Then one can still use either an FE or RE formulation to fit the model and interpret the results, as discussed in preceding sections.

A second standard method is to use an autoregressive model with lagged dependent variables as regressor to capture dynamic dependence. The lagged dependent variable is treated as a shifter of the current transition probability after controlling for other regressors and individual unobserved heterogeneity. If there are compelling reasons to model dynamics by introducing y_{it-1} as a regressor, then estimation is straightforward under RE assumptions. The presence of a lagged dependent variable implies that the effect of any shock has an impact on outcome in all future periods. Hence, a distinction between contemporaneous MES and dynamic MES arises in principle. There is scant literature that discusses this issue in the context of discrete outcomes.

An FE logit model with lagged dependent variable adds y_{it-1} as a regressor in (22.6). Conceptually, this model is attractive because it allows for distinction between unobserved heterogeneity (α_i) and true state dependence (y_{it-1}). Estimation is challenging because the usual tricks for eliminating fixed effects (and time-invariant regressors) do not work.

Initial work by [Honoré and Kyriazidou \(2000\)](#) considered a pure autoregressive panel model. [Bartolucci and Nigro \(2010\)](#) considered a more general regression model with a lagged dependent variable that is not equivalent to the logit model but mimics it and hence may be treated as an alternative. This alternative model, referred to as the conditional quadratic exponential specification, adds interaction terms of form $y_{it-1}y_{iT}$ (hence the term “quadratic”), which enables elimination of the α_i by conditioning on $\sum_t y_{it}$. It uses only the subset of observations in the sample for which at least one transition is recorded. The community-contributed `cquad` package ([Bartolucci 2015](#)) implements several variants of this estimator.

An alternative method is to obtain bias-corrected estimates from regular logit regression of y_{it} on individual-specific DVs and regressors that include the lagged dependent variable. The `logitfe` command ([Cruz-Gonzalez, Fernández-Val, and Weidner 2017](#)) also covers this case and can provide both parameter estimates and AMES. In the dynamic case, one needs to use `lags (#)` to specify the value of a necessary trimming parameter.

22.4.14 Panel probit models

Panel probit models can be fit using the `xtprobit` command, which is similar to the `xtlogit` command and has similar options.

An important difference, however, is that the `fe` option is not available because, unlike for the panel logit model, conditioning on $\sum_{t=1}^T y_{it}$ does not eliminate the

fixed effects. Recent work has pointed out the often-referenced result that consistent estimation of the slope coefficients of the probit FE model in a short panel is restricted to the case $T = 2$. It is possible that consistent estimation procedures will be developed for $T > 2$.

For FE models, the easiest way to proceed is to obtain bias-corrected estimates of a panel probit DV model using the `probitfe` command ([Cruz-Gonzalez, Fernández-Val, and Weidner 2017](#)) or to fit a probit CRE model. Both estimators can also yield AMES.

22.4.15 Panel ordered logit and probit models

The outcome variable sometimes may be categorical and ordered. Measuring outcomes on an ordinal scale is common, especially for outcomes that are measured subjectively. For example, a patient may be asked to report pain intensity level on a 10-point scale or personal health status according to five categories such as poor, fair, good, very good, and excellent. When such responses are obtained in repeated samples from the same individuals, ordered logit or probit models provide a natural framework for analyzing such multinomial outcomes; see section [18.9](#).

Stata provides two commands, `xtologit` and `xtoprobit`, for RE estimation of ordered categorical panel data. The syntax of these commands is similar to that for the logit and probit panel models.

We illustrate the `xtologit` command by analyzing recoded Rand count data on annual doctor visits (`mdu`). The 7 recoded categories are as follows: 0 (no visits), 1 (1 or 2 visits), 2 (3 or 4 visits), 3 (5 or 6 visits), 4 (7 or 8 visits), 5 (8 to 12 visits), and 6 (more than 12 visits). This categorization is somewhat arbitrary and only illustrative.

The following descriptive summary displays some key features of the outcome variable.

```

. * Tabulate cmdu generated by recoding count into seven ordered categories
. recode mdu (0 = 1) (1/2 = 2) (3/4 = 3) (5/6 = 4) (7/8 = 5)
> (8/12 = 6) (13/999 = 7), gen(cmdu)
(15507 differences between mdu and cmdu)

. tabulate cmdu

```

RECODE of mdu (Number face-to-fac t md visits)	Freq.	Percent	Cum.
1	6,308	31.25	31.25
2	6,610	32.75	63.99
3	3,229	16.00	79.99
4	1,657	8.21	88.20
5	939	4.65	92.85
6	801	3.97	96.82
7	642	3.18	100.00
Total	20,186	100.00	

The first three categories account for nearly 80% of the probability mass. The more-than-12-visits category is the smallest, accounting for just over 3% of the probability mass.

The `xtlogit` command is next executed with the cluster-robust option for standard errors.

```

. * RE ordered logit: Estimates
. xtologit cmdu lcoins ndisease female age lfam child, vce(cluster id) nolog
Random-effects ordered logistic regression
Group variable: id
Random effects u_i ~ Gaussian
Integration method: mvaghermite
Log pseudolikelihood = -29148.65
Number of obs      = 20,186
Number of groups  = 5,908
Obs per group:
    min =       1
    avg =     3.4
    max =       5
Integration pts. =     12
Wald chi2(6)      = 766.74
Prob > chi2       = 0.0000
(Std. err. adjusted for 5,908 clusters in id)

```

cmdu	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
lcoins	-.2304359	.0144439	-15.95	0.000	-.2587454	-.2021265
ndisease	.0832606	.004843	17.19	0.000	.0737685	.0927527
female	.4172223	.0602473	6.93	0.000	.2991398	.5353048
age	.0064605	.0028157	2.29	0.022	.0009418	.0119793
lfam	-.3604544	.0588079	-6.13	0.000	-.4757157	-.2451931
child	.2004983	.0863054	2.32	0.020	.0313428	.3696537
/cut1	-.9207318	.1411173			-1.197426	-.6440378
/cut2	1.312918	.1416307			1.035327	1.590509
/cut3	2.612945	.1428847			2.332896	2.892994
/cut4	3.581754	.1444939			3.298551	3.864957
/cut5	4.407798	.1470729			4.119541	4.696056
/cut6	5.627819	.1531991			5.327555	5.928084
/sigma2_u	3.666984	.1353704			3.411034	3.942139

The individual-specific random component of the intercept follows the normal distribution whose estimated variance is reported in the last row of the output. The log likelihood does not have closed expression, so numerical integration is used in optimization. The regression coefficients indicate qualitatively the same signs as in the RE panel count models reported in later sections of this chapter. We focus on the key variable `lcoins`, the log of the coinsurance rate. The result indicates that a higher coinsurance rate is associated with fewer visits on average. As in the case of ordered logit, the output includes the six estimated cutoffs.

The regression coefficients are less informative than the MES. Next we use the `margins` command to obtain the AMES of `lcoins`.

```

. * RE ordered logit: AMEs of coinsurance rate on probability of category outcomes
. margins, dydx(lcoins)
Average marginal effects                                         Number of obs = 20,186
Model VCE: Robust
dy/dx wrt: lcoins
1._predict: Pr(1.cmdu), predict(pr outcome(1))
2._predict: Pr(2.cmdu), predict(pr outcome(2))
3._predict: Pr(3.cmdu), predict(pr outcome(3))
4._predict: Pr(4.cmdu), predict(pr outcome(4))
5._predict: Pr(5.cmdu), predict(pr outcome(5))
6._predict: Pr(6.cmdu), predict(pr outcome(6))
7._predict: Pr(7.cmdu), predict(pr outcome(7))

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
lcoins						
-predict						
1	.0301368	.0018442	16.34	0.000	.0265223	.0337513
2	.0017421	.0003038	5.73	0.000	.0011467	.0023376
3	-.0079852	.0005111	-15.62	0.000	-.0089869	-.0069834
4	-.0072376	.0004694	-15.42	0.000	-.0081575	-.0063176
5	-.0054922	.0003726	-14.74	0.000	-.0062226	-.0047619
6	-.0057985	.0004114	-14.09	0.000	-.0066049	-.0049921
7	-.0053655	.0004143	-12.95	0.000	-.0061775	-.0045534

Computing MES is somewhat slower than model estimation; it involves numerical integration because, as explained in section [22.4.11](#), the ME depends upon the distribution of the individual-specific random effects. The estimated AME is obtained by averaging over the distribution of the random effects.

The estimated AME, the change in the probability of outcome being in a particular category, varies by category. The effect is positive in the first two categories with two or fewer visits; the effect is relatively small and negative, but highly significant, in categories with three or more visits. This result illustrates that in an ordered model, the sign of the regressor does not necessarily reflect the sign of the ME in all categories. As expected, the AMEs sum to zero across the seven categories.

Finally, we use the `predict` option to compare the fitted distribution with actual frequency distribution of outcomes.

```

. * RE ordered logit: Estimate fitted average category probabilities
. predict pr*, pr
(using 12 quadrature points)
. summarize pr*

```

Variable	Obs	Mean	Std. dev.	Min	Max
pr1	20,186	.3142163	.1071772	.0116773	.62827
pr2	20,186	.3187694	.0263436	.0582258	.3359595
pr3	20,186	.1592766	.0282325	.0702338	.1995985
pr4	20,186	.0851783	.0264847	.0257301	.149432
pr5	20,186	.0495467	.0212988	.0112889	.1275895
pr6	20,186	.0424561	.0243255	.0073362	.1875392
pr7	20,186	.0305566	.0272974	.0035658	.4295574

The result shows that the fitted distribution of probability mass across categories is quite similar to that in the data displayed at the start of this section.

The analysis of the same data using `xtpoprobit` is left to the interested reader.

22.5 Tobit and interval-data models

We fit a panel tobit model for medical expenditures (`med`). Then the only panel estimator available is the RE estimator, which introduces a normally distributed individual-specific effect. Consistent estimation of an FE model is possible only if $T \rightarrow \infty$.

22.5.1 Panel summary of the dependent variable

For simplicity, we model expenditures in levels, though from section 19.4, the key assumption of normality for the tobit model is more reasonable for the natural logarithm of expenditures.

The dependent variable, `med`, has within variation and between variation of similar magnitude because the `xtsum` command yields

```
. * Tobit: Panel summary of dependent variable
. xtsum med
```

Variable		Mean	Std. dev.	Min	Max	Observations
med	overall	171.5892	698.2689	0	39182.02	N = 20186
	between		503.2589	0	19615.14	n = 5908
	within		526.269	-19395.28	20347.2	T-bar = 3.41672

22.5.2 RE tobit model

The RE panel tobit model specifies the latent variable y_{it}^* to depend on regressors, an idiosyncratic error, and an individual-specific error, so

$$y_{it}^* = \mathbf{x}'_{it}\beta + \alpha_i + \varepsilon_{it} \quad (22.10)$$

where $\alpha_i \sim N(0, \sigma_\alpha^2)$ and $\varepsilon_{it} \sim N(0, \sigma_\varepsilon^2)$ and the regressor vector \mathbf{x}_{it} includes an intercept. For left-censoring at L , we observe the y_{it} variable, where

$$(22.11)$$

$$y_{it} = \begin{cases} y_{it}^* & \text{if } y_{it}^* > L \\ L & \text{if } y_{it}^* \leq L \end{cases}$$

22.5.3 The xttobit command

The `xttobit` command has a similar syntax to the cross-sectional `tobit` command. The `ll()` option is used to define the lower limit for left-censoring, and the `ul()` option is used to define the upper limit for right-censoring. The limit can be a variable, not just a number, so more generally we can have the limit L_i rather than the limit L in (22.11). Like the RE logit model, estimation requires univariate numerical integration, using Gauss–Hermite quadrature. The `vce(bootstrap)` option can provide panel-robust standard errors.

For our data, we obtain

```

. * Tobit RE estimator
. xttobit med lcoins ndisease female age lfam child, ll(0) nolog

Random-effects tobit regression                               Number of obs     = 20,186
                                                               Uncensored = 15,733
                                                               Left-censored = 4,453
                                                               Right-censored = 0

Limits: Lower =      0
          Upper = +inf

Group variable: id                                         Number of groups = 5,908
Random effects u_i ~ Gaussian                            Obs per group:
                                                               min =        1
                                                               avg =      3.4
                                                               max =        5

Integration method: mvaghermite                           Integration pts. =    12
                                                               Wald chi2(6) = 573.45
                                                               Prob > chi2 = 0.0000

Log likelihood = -130030.45

```

med	Coefficient	Std. err.	z	P> z	[95% conf. interval]
lcoins	-31.10247	3.578498	-8.69	0.000	-38.1162 -24.08875
ndisease	13.49452	1.139156	11.85	0.000	11.26182 15.72722
female	60.10112	14.95966	4.02	0.000	30.78072 89.42152
age	4.075582	.7238253	5.63	0.000	2.656911 5.494254
lfam	-57.75023	14.68422	-3.93	0.000	-86.53077 -28.96968
child	-52.02314	24.21619	-2.15	0.032	-99.48599 -4.560284
_cons	-98.27203	36.05977	-2.73	0.006	-168.9479 -27.59618
/sigma_u	371.3134	8.64634	42.94	0.000	354.3668 388.2599
/sigma_e	715.1779	4.704581	152.02	0.000	705.9571 724.3987
rho	.2123246	.0086583		.1957541	.2296872

LR test of sigma_u=0: chibar2(01) = 778.18 Prob >= chibar2 = 0.000

About 22% of the observations are censored (4,453 of 20,186). All regressor coefficients are statistically significant and have the expected sign. The estimated standard deviation of the random effects α_i is 371.3 and is highly statistically significant. The quantity labeled `rho` equals $\sigma_\alpha^2 / (\sigma_\alpha^2 + \sigma_\varepsilon^2)$ and measures the fraction of the total variance, $\sigma_\alpha^2 + \sigma_\varepsilon^2$, that is due to the random effects. In an exercise, we compare these estimates with those from the `tobit` command, which treats observations as independent over i and t (so $\alpha_i = 0$). The estimates are similar.

22.5.4 The xheckman command

The `xheckman` command extends the Heckman selection model `heckman` command to the panel case by adding individual-specific random errors to the outcome and selection equations.

The cross-sectional selection model is presented in section [19.6](#). In the panel case, we specify the latent variables for selection and outcome to be, respectively,

$$\begin{aligned}y_{1it}^* &= \mathbf{x}'_{1it}\boldsymbol{\beta}_1 + \alpha_{1i} + \varepsilon_{1it} \\y_{2it}^* &= \mathbf{x}'_{2it}\boldsymbol{\beta}_2 + \alpha_{2i} + \varepsilon_{2it}\end{aligned}$$

The outcome y_{2it} is observed only when $y_{1it}^* > 0$. The individual-specific errors α_{1i} and α_{2i} are specified to be bivariate normally distributed, and the idiosyncratic errors ε_{1it} and ε_{2it} are specified to be bivariate normally distributed with the normalization that $\text{Var}(\varepsilon_{1it}) = 1$. As in the cross-sectional case, it is desirable for some variables in \mathbf{x}_{1it} to differ from those in \mathbf{x}_{2it} to reduce the reliance on strong distributional assumptions.

ML estimates can be obtained using the `xheckman` command, which has a similar syntax to the cross-sectional `heckman` command. Estimation uses a computational algorithm related to [Roodman \(2011\)](#). There is no closed-form solution for the log likelihood, so estimation is by Gaussian quadrature. The `intpoints(#)` option allows many more integration points than the default of seven points. A higher number leads to greater precision and can lead to greater likelihood of convergence but adds to computational time. Convergence can be challenging if the identification of some of the parameters is poor or due to failure of the parametric assumptions.

As an illustration, we model outpatient medical expenditures (`outpdol`); convergence problems arose using variable `med`, which is much more highly skewed because it additionally includes inpatient expenditures. To aid convergence, we make the strong (and questionable) exclusion restrictions that variables `female`, `lfam`, `child`, `hlthf` (health status fair), `hlthp` (health status poor), and `linc` (log family income) do not appear in the outcome model. We set `intpoints(15)` and, to speed computation, use a 10% subsample. We obtain

```

. * Heckman sample-selection panel estimator
. gen doutpdol = outpdol > 0
. xtheckman outpdol lcoins ndisease age if id > 629388, intpoints(10) nolog
>      select(doutpdol = lcoins ndisease age female lfam child hlthf hlthp linc)
Random-effects regression with selection
Number of obs      =  2,016
Selected          =  1,286
Nonselected       =   730
Group variable: id
Number of groups  =    638
Obs per group:
min              =      1
avg              =   3.2
max              =      5
Integration method: mvaghermite
Integration pts.  =     10
Wald chi2(3)      =  58.44
Prob > chi2       = 0.0000
Log likelihood = -8425.1472

```

	Coefficient	Std. err.	z	P> z	[95% conf. interval]
outpdol					
lcoins	.5330196	1.358528	0.39	0.695	-2.129646 3.195685
ndisease	1.9796	.4178257	4.74	0.000	1.160676 2.798523
age	.6824011	.1496004	4.56	0.000	.3891897 .9756125
_cons	29.81417	7.534019	3.96	0.000	15.04776 44.58057
doutpdol					
lcoins	-.2035518	.0284604	-7.15	0.000	-.2593331 -.1477704
ndisease	.0598368	.010309	5.80	0.000	.0396315 .080042
age	-.0067661	.0057241	-1.18	0.237	-.0179851 .004453
female	.088402	.1128548	0.78	0.433	-.1327894 .3095934
lfam	-.4223696	.1147663	-3.68	0.000	-.6473075 -.1974318
child	-.5411109	.177272	-3.05	0.002	-.8885576 -.1936642
hlthf	.0395699	.1946469	0.20	0.839	-.3419311 .4210708
hlthp	.1131886	.3852884	0.29	0.769	-.6419627 .8683399
linc	.1146879	.0343245	3.34	0.001	.0474131 .1819627
_cons	.4863538	.410284	1.19	0.236	-.3177881 1.290496
var(e.outpdol)	4286.03	271.7522			3785.17 4853.163
corr(e.dout~1, e.outpdol)	-.6172399	.1096029	-5.63	0.000	-.7885422 -.3570515
var(out~1[id])	1549.2	207.8337			1191.007 2015.12
var(dou~1[id])	.9505599	.1499978			.6976866 1.295086
corr(doutpdol[id], outpdol[id])	.4910298	.134623	3.65	0.000	.1874851 .7089668

The two correlation coefficients are quite precisely estimated, and they imply presence of endogenous selection. The individual-specific errors (α_{1i}, α_{2i}) are positively correlated with correlation 0.49. The idiosyncratic errors ($\varepsilon_{1it}, \varepsilon_{2it}$) are negatively correlated.

Additional output generated from this model can assist in further interpretation. We use the `margins` command to generate predicted mean outpatient expenditures, conditional on outpatient expenditures being positive, at two different ages with the other predictors evaluated at sample mean values.

```
. * Predictive margins for outpdol given outpdol > 0 at different ages
. margins, at(age=(20(40)60)) predict(ycond)

Predictive margins                                         Number of obs = 2,016
Model VCE: OIM
Expression: mean of outpdol, predict(ycond)
1._at: age = 20
2._at: age = 60
```

	Delta-method					[95% conf. interval]
	Margin	std. err.	z	P> z		
<code>_at</code>						
	1	56.58816	2.77992	20.36	0.000	51.13961
	2	82.24204	6.621706	12.42	0.000	69.26373
						95.22034

22.5.5 The `xtintreg` and `xtfrontier` commands

The `xtintreg` command estimates the parameters of interval-data models where continuous data are reported only in ranges. For example, annual medical expenditure data may be reported only as \$0, between \$0 and \$100, between \$100 and \$1,000, and more than \$1,000. The unobserved continuous variable, y_{it}^* , is modeled as in (22.10), and the observed variable, y_{it} , arises as y_{it}^* falls into the appropriate range.

Stochastic production frontier models introduce into the production function a strictly negative error term that pushes production below the efficient level. In the simplest panel model, this error term is time invariant and has a truncated normal distribution, so the model has some

commonalities with the panel tobit model. The `xtfrontier` command is used to estimate the parameters of these models.

All four commands—`xttobit`, `xtheckman`, `xtintreg`, and `xtfrontier`—rely heavily on the assumption of homoskedastic normally distributed errors for consistency and, like their cross-sectional counterparts, are more fragile to distributional misspecification than, for example, linear models and logit models.

22.6 Count-data models

We fit count models for the number of doctor visits (`mdu`). Many of the relevant issues have already been raised for the `xtlogit` command. One difference is that analytical solutions are possible for count RE models, by appropriate choice of (nonnormal) distribution for the random effects. A second difference is that Poisson panel estimators have the same robustness properties as Poisson cross-sectional estimators. They are consistent even if the data are not Poisson distributed, provided the conditional mean is correctly specified. At the same time, count data are often overdispersed, and the need to use heteroskedasticity-robust standard errors in the cross-sectional case carries over to a need to use panel-robust standard errors in the panel case.

22.6.1 The `xtpoisson` command

The `xtpoisson` command has the syntax

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

The options include PA (`pa`), RE (`re` and `normal`), and FE (`fe`) models.

Then PA, RE, and FE estimators are available for the Poisson model, with the `xtpoisson` command. Panel estimators are also available for the negative binomial model, with `xtnbreg`.

22.6.2 Panel summary of the dependent variable

The dependent variable, `mdu`, is considerably overdispersed because, from section [22.3.1](#), the sample variance of $4.50^2 = 20.25$ is 7 times the sample mean of 2.86. This makes it very likely that default standard errors for both cross-sectional and panel Poisson estimators will considerably underestimate the true standard errors.

The `mdu` variable has a within variation of magnitude similar to the between variation.

```
. * Poisson: Panel summary of dependent variable
. xtsum mdu
```

Variable		Mean	Std. dev.	Min	Max	Observations
mdu	overall	2.860696	4.504765	0	77	N = 20186
	between		3.785971	0	63.33333	n = 5908
	within		2.575881	-34.47264	40.0607	T-bar = 3.41672

To provide more detail on the variation in `mdu` over time, we look at transition probabilities, after first aggregating all instances of four or more doctor visits into a single category. We have

```
. * Year-to-year transitions in doctor visits
. generate mdushort = mdu
. replace mdushort = 4 if mdu >= 4
(4,039 real changes made)
. xttrans mdushort
```

mdushort	mdushort					Total
	0	1	2	3	4	
0	58.87	19.61	9.21	4.88	7.42	100.00
1	33.16	24.95	17.58	10.14	14.16	100.00
2	23.55	24.26	17.90	12.10	22.19	100.00
3	17.80	20.74	18.55	12.14	30.77	100.00
4	8.79	11.72	12.32	11.93	55.23	100.00
Total	31.81	19.27	13.73	9.46	25.73	100.00

There is considerable persistence: over half of people with zero doctor visits one year also have zero visits the next year, and over half of people with four or more visits one year also have four or more visits the next year.

We compute the correlations over time in the dependent variable.

```
. * Correlations over time in the dependent variable
. pwcorr mdu L1.mdu L2.mdu L3.mdu L4.mdu
```

	mdu	L.mdu	L2.mdu	L3.mdu	L4.mdu
mdu	1.0000				
L.mdu	0.6184	1.0000			
L2.mdu	0.4744	0.6029	1.0000		
L3.mdu	0.3714	0.4602	0.5995	1.0000	
L4.mdu	0.3820	0.3702	0.5054	0.6100	1.0000

These correlations are dampening over time. Thus, the PA estimator used below will relax the assumption of equicorrelation used in the logit example to allow more flexible correlation.

22.6.3 Pooled Poisson estimator

The pooled Poisson estimator assumes that y_{it} is Poisson distributed with a mean of

$$E(y_{it} | \mathbf{x}_{it}) = \exp(\mathbf{x}'_{it}\boldsymbol{\beta}) \quad (22.12)$$

as in the cross-sectional case. Consistency of this estimator requires that (22.12) be correctly specified but does not require that the data actually be Poisson distributed. If the data are not Poisson distributed, however, then it is essential that robust standard errors be used.

The pooled Poisson estimator can be estimated by using the `poisson` command, with cluster-robust standard errors that take care of both overdispersion and serial correlation. We have

```

. * Pooled Poisson estimator with cluster--robust standard errors
. poisson mdu lcoins ndisease female age lfam child, vce(cluster id)
Iteration 0: log pseudolikelihood = -62580.248
Iteration 1: log pseudolikelihood = -62579.401
Iteration 2: log pseudolikelihood = -62579.401

Poisson regression
Number of obs = 20,186
Wald chi2(6) = 476.93
Prob > chi2 = 0.0000
Pseudo R2 = 0.0609
Log pseudolikelihood = -62579.401
(Std. err. adjusted for 5,908 clusters in id)

```

mdu	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]	
lcoins	-.0808023	.0080013	-10.10	0.000	-.0964846	-.0651199
ndisease	.0339334	.0026024	13.04	0.000	.0288328	.039034
female	.1717862	.0342551	5.01	0.000	.1046473	.2389251
age	.0040585	.0016891	2.40	0.016	.000748	.0073691
lfam	-.1481981	.0323434	-4.58	0.000	-.21159	-.0848062
child	.1030453	.0506901	2.03	0.042	.0036944	.2023961
_cons	.748789	.0785738	9.53	0.000	.5947872	.9027907

The importance of using cluster-robust standard errors cannot be overemphasized. For these data, the correct cluster-robust standard errors are 50% higher than the heteroskedasticity-robust standard errors and 300% higher than the default standard errors; see the end-of-chapter exercises. Here failure to control for autocorrelation and failure to control for overdispersion both lead to considerable understatement of the true standard errors.

22.6.4 PA Poisson estimator

The PA Poisson estimator is a variation of the pooled Poisson estimator that relaxes the assumption of independence of y_{it} to allow different models for the correlation

$$\rho_{ts} = \text{Cor}[\{y_{it} - \exp(\mathbf{x}'_{it}\boldsymbol{\beta})\}\{y_{is} - \exp(\mathbf{x}'_{is}\boldsymbol{\beta})\}]$$

This estimator is obtained by using the `xtpoisson` command with the `pa` option. Different correlation models are specified by using the `corr()`

option; see section 8.4.3. Consistency of this estimator requires only that (22.12) be correct. But if the data are non-Poisson and are overdispersed, then the `vce(robust)` option should be used because otherwise default standard errors will underestimate the true standard errors.

We use the `corr(unstructured)` option so that ρ_{ts} can vary freely over t and s . We obtain

```
. * Poisson PA estimator with unstructured error correlation and robust VCE
. xtpoisson mdu lcoins ndisease female age lfam child, pa corr(unstr) vce(robust)

Iteration 1: tolerance = .01585489
Iteration 2: tolerance = .00034066
Iteration 3: tolerance = 2.334e-06
Iteration 4: tolerance = 1.939e-08

GEE population-averaged model
Number of obs      = 20,186
Group and time vars: id year
Number of groups   = 5,908
Family: Poisson
Obs per group:
Link: Log          min =      1
Correlation: unstructured avg =    3.4
                                         max =      5
                                         Wald chi2(6)     = 508.61
Scale parameter = 1             Prob > chi2     = 0.0000
                                         (Std. err. adjusted for clustering on id)
```

mdu	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
lcoins	-.0804454	.0077782	-10.34	0.000	-.0956904	-.0652004
ndisease	.0346067	.0024238	14.28	0.000	.0298561	.0393573
female	.1585075	.0334407	4.74	0.000	.0929649	.2240502
age	.0030901	.0015356	2.01	0.044	.0000803	.0060999
lfam	-.1406549	.0293672	-4.79	0.000	-.1982135	-.0830962
child	.1013677	.04301	2.36	0.018	.0170696	.1856658
_cons	.7764626	.0717221	10.83	0.000	.6358897	.9170354

The coefficient estimates are quite similar to those from pooled Poisson. The standard errors are as much as 10% lower, reflecting efficiency gain due to better modeling of the correlations.

The estimated autocorrelation matrix is stored in `e(R)`. We have

```

. * Correlations over time
. matrix list e(R)
symmetric e(R) [5,5]
      c1          c2          c3          c4          c5
r1       1
r2   .53143297      1
r3   .40817495   .58547795      1
r4   .32357326   .35321716   .54321752      1
r5   .34152288   .29803555   .43767583   .61948751      1

```

The correlations do not drop greatly as the lag length increases, so the simpler equicorrelation may be a reasonable approximation.

A more detailed comparison of estimators and methods to estimate the VCE (see the end-of-chapter exercises) shows that failure to use the `vce(robust)` option leads to erroneous standard errors that are one-third of the robust standard errors, and similar estimates are obtained by using the `corr(exchangeable)` or `corr(ar2)` option.

22.6.5 RE Poisson estimators

The Poisson individual-effects model assumes that y_{it} is Poisson distributed with a mean of

$$E(y_{it}|\alpha_i, \mathbf{x}_{it}) = \exp(\gamma_i + \mathbf{x}'_{it}\boldsymbol{\beta}) = \alpha_i \exp(\mathbf{x}'_{it}\boldsymbol{\beta}) \quad (22.13)$$

where $\gamma_i = \ln \alpha_i$, and here \mathbf{x}_{it} includes an intercept. The conditional mean can be viewed either as one with effects that are additive before exponentiation or as one with multiplicative effects.

The standard Poisson RE estimator assumes that α_i is gamma distributed with a mean of 1 and a variance of η . This assumption has the attraction that there is a closed-form expression for the integral (22.4), so the estimator is easy to compute. Furthermore, then $E(y_{it}|\mathbf{x}_{it}) = \exp(\mathbf{x}'_{it}\boldsymbol{\beta})$ so that predictions and MES are easily obtained and interpreted. This is the conditional mean given in (22.12) for the PA and pooled models, so for the special case of the Poisson, and unlike the logit model, the PA, pooled, and

RE estimators of β have the same probability limit. Finally, the first-order conditions for the Poisson RE estimator, $\hat{\beta}$, can be shown to be

$$\sum_{i=1}^N \sum_{t=1}^T \mathbf{x}_{it} \left(y_{it} - \lambda_{it} \frac{\bar{y}_i + \eta/T}{\bar{\lambda}_i + \eta/T} \right) = \mathbf{0} \quad (22.14)$$

where $\lambda_{it} = \exp(\mathbf{x}'_{it}\beta)$ and $\bar{\lambda}_i = T^{-1} \sum_t \exp(\mathbf{x}'_{it}\beta)$, so the estimator is consistent if $E(y_{it}|\alpha_i, \mathbf{x}_{i1}, \dots, \mathbf{x}_{iT}) = \alpha_i \exp(\mathbf{x}'_{it}\beta)$ because then the left-hand side of (22.14) has an expected value of 0.

The RE estimator is obtained by using the `xtpoisson` command with the `re` option. We use the `vce(robust)` option to obtain panel-robust standard errors. We have

```
. * Poisson RE estimator with cluster--robust standard errors
. xtpoisson mdu lcoins ndisease female age lfam child, re vce(robust) nolog
Random-effects Poisson regression
                                         Number of obs      =  20,186
Group variable: id                      Number of groups =  5,908
                                         Obs per group:
                                         min =        1
                                         avg =      3.4
                                         max =       5
                                         Wald chi2(6)    = 5407.91
Log pseudolikelihood = -43240.556          Prob > chi2     = 0.0000
                                         (Std. err. adjusted for clustering on id)
```

mdu	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
lcoins	-.0878258	.0079239	-11.08	0.000	-.1033563 -.0722952
ndisease	.0387629	.0024087	16.09	0.000	.0340421 .0434838
female	.1667192	.0345869	4.82	0.000	.09893 .2345083
age	.0019159	.0016533	1.16	0.247	-.0013244 .0051563
lfam	-.1351786	.0360629	-3.75	0.000	-.2058606 -.0644966
child	.1082678	.0533869	2.03	0.043	.0036314 .2129043
_cons	.7574177	.0831229	9.11	0.000	.5944999 .9203355
/lnalpha	.0251256	.0905423			-.1523339 .2025852
alpha	1.025444	.092846			.8587015 1.224564

LR test of alpha=0: chibar2(01) = 3.9e+04

Prob >= chibar2 = 0.000

Compared with the PA estimates, the RE coefficients are within 10%, and the RE cluster-robust standard errors are about 10% higher. The cluster-robust standard errors for the RE estimates are 20–50% higher than the default standard errors, not reported, so cluster-robust standard errors are needed. The problem is that the Poisson RE model is not sufficiently flexible because the single additional parameter, η , needs to simultaneously account for both overdispersion and correlation. Cluster-robust standard errors can correct for this, or the richer negative binomial RE model may be used.

An alternative Poisson RE estimator assumes that $\gamma_i = \ln \alpha_i$ is normally distributed with a mean of 0 and a variance of σ_α^2 , similar to the `xtlogit` and `xtprobit` commands. Here estimation is much slower because Gauss-Hermite quadrature is used to perform numerical univariate integration. And similarly to the logit RE estimator, prediction and computation of MES is difficult. This alternative Poisson RE estimator can be computed by using `xtpoisson` with the `normal` option. Estimates from this method are presented in section [22.6.7](#).

The RE model permits only the intercept to be random. We can also allow slope coefficients to be random. This is the mixed-effects Poisson estimator implemented with `mepoisson`. The method is similar to that for `melogit`, presented in section [22.4.12](#). The method is computationally intensive.

22.6.6 FE Poisson estimator

The FE model is the Poisson individual-effects model ([22.13](#)), where α_i is now possibly correlated with \mathbf{x}_{it} , and in short panels, we need to eliminate α_i before estimating β .

These effects can be eliminated by using the conditional ML estimator based on a log density for the i th individual that conditions on $\sum_{t=1}^{T_i} y_{it}$, similar to the treatment of fixed effects in the logit model. Some algebra leads to the Poisson FE estimator with first-order conditions

$$\sum_{i=1}^N \sum_{t=1}^T \mathbf{x}_{it} \left(y_{it} - \frac{\lambda_{it}}{\bar{\lambda}_i} \bar{y}_i \right) = \mathbf{0} \quad (22.15)$$

where $\lambda_{it} = \exp(\mathbf{x}'_{it}\boldsymbol{\beta})$ and $\bar{\lambda}_i = T^{-1} \sum_t \exp(\mathbf{x}'_{it}\boldsymbol{\beta})$. The Poisson FE estimator is therefore consistent if $E(y_{it}|\alpha_i, \mathbf{x}_{i1}, \dots, \mathbf{x}_{iT}) = \alpha_i \exp(\mathbf{x}'_{it}\boldsymbol{\beta})$ because then the left-hand side of (22.15) has the expected value of zero.

The Poisson FE estimator can be obtained by using the `xtpoisson` command with the `fe` option. To obtain cluster-robust standard errors, we can use the `vce(robust)` option. We have

```
. * Poisson FE estimator with cluster--robust standard errors
. xtpoisson mdu lcoins ndisease female age lfam child, fe vce(robust)
note: 265 groups (265 obs) dropped because of only one obs per group
note: 666 groups (2130 obs) dropped because of all zero outcomes
note: lcoins dropped because it is constant within group
note: ndisease dropped because it is constant within group
note: female dropped because it is constant within group

Iteration 0:  log pseudolikelihood = -24182.852
Iteration 1:  log pseudolikelihood = -24173.211
Iteration 2:  log pseudolikelihood = -24173.211

Conditional fixed-effects Poisson regression
Number of obs      = 17,791
Group variable: id
Number of groups  = 4,977
Obs per group:
min =          2
avg =          3.6
max =          5
Wald chi2(3)      =   4.58
Log pseudolikelihood = -24173.211
Prob > chi2       = 0.2051
(Std. err. adjusted for clustering on id)
```

mdu	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
age	-.0112009	.0091493	-1.22	0.221	-.0291331	.0067314
lfam	.0877134	.1160837	0.76	0.450	-.1398064	.3152332
child	.1059867	.0786326	1.35	0.178	-.0481304	.2601037

Only the coefficients of time-varying regressors are identified, similar to other FE model estimators. The Poisson FE estimator requires that there be at least two periods of data, leading to a loss of 265 observations, and that the count for an individual be nonzero in at least one period ($\sum_{t=1}^{T_i} y_{it} > 0$), leading to a loss of 666 individuals because `mdu` equals 0 in all periods for 666 people. The cluster-robust standard errors are roughly two times those of the default standard errors; see the end-of-chapter exercises. In theory, the

individual effects, α_i , could account for overdispersion, but as is common, they do not completely do so. The standard errors are also roughly twice as large as the PA and RE standard errors, reflecting a loss of precision due to using only within variation.

For the FE model, results should be interpreted on the basis of $E(y_{it}|\alpha_i, \mathbf{x}_{it}) = \alpha_i \exp(\mathbf{x}'_{it}\boldsymbol{\beta})$. The `predict` command with the `nu0` option gives predictions when $\gamma_i = 0$ so $\alpha_i = 1$, and the `margins, dydx()` command with the `predict(nu0)` option gives the corresponding MES. If we do not want to consider only the case of $\alpha_i = 1$, then the model implies that $\partial E(y_{it}|\alpha_i, \mathbf{x}_{it})/\partial x_{j,it} = \beta_j \times E(y_{it}|\alpha_i, \mathbf{x}_{it})$, so β_j can still be interpreted as a semielasticity.

Given the estimating equations given by (22.15), the Poisson FE estimator can be applied to any model with multiplicative effects and an exponential conditional mean, essentially whenever the dependent variable has a positive conditional mean. Then the Poisson FE estimator uses the quasi-difference, $y_{it} - (\lambda_{it}/\bar{\lambda}_i)\bar{y}_i$, whereas the linear model uses the mean difference, $y_{it} - \bar{y}_i$.

In the linear model, one can instead use the first difference, $y_{it} - y_{i,t-1}$, to eliminate the fixed effects, and this has the additional advantage of enabling estimation of FE dynamic linear models using the Arellano–Bond estimator. Similarly, here one can instead use the alternative quasi-difference, $(\lambda_{i,t-1}/\lambda_{it})y_{it} - y_{i,t-1}$, to eliminate the fixed effects and use this as the basis for estimation of dynamic panel count models; for details see, for example, [Cameron and Trivedi \(2013, chap. 9\)](#).

The correlated RE model, presented in section 22.2.4, provides an alternative way to control for FE. Estimates for this model are presented below.

22.6.7 Panel Poisson estimators comparison

We compute several panel Poisson estimators, with panel-robust VCE used for all estimators.

```
. * Compute various Poisson panel estimators
. global xlist age lfam child lcoins ndisease female
. qui xtpoisson mdu $xlist, pa corr(unstr) vce(robust)
. estimates store PPA_ROB
. qui xtpoisson mdu $xlist, re vce(robust)
. estimates store PRE
. qui xtpoisson mdu $xlist, re normal vce(robust)
. estimates store PRE_NORM
. qui xtpoisson mdu $xlist, fe vce(robust)
. estimates store PFE
. qui xtpoisson mdu $xlist aveage avelfam avechild, re normal vce(robust)
. estimates store CRE
```

The estimates are

```
. * Report various Poisson panel estimators - results
. estimates table PPA_ROB PRE PRE_NORM PFE CRE, equations(1) b(%8.4f)
>     se stats(N ll) stfmt(%8.0f)
```

Variable	PPA_ROB	PRE	PRE_NORM	PFE	CRE
#1					
age	0.0031 0.0015	0.0019 0.0017	0.0027 0.0017	-0.0112 0.0091	-0.0112 0.0092
lfam	-0.1407 0.0294	-0.1352 0.0361	-0.1443 0.0359	0.0877 0.1161	0.0872 0.1158
child	0.1014 0.0430	0.1083 0.0534	0.0737 0.0534	0.1060 0.0786	0.1059 0.0786
lcoins	-0.0804 0.0078	-0.0878 0.0079	-0.1145 0.0072		-0.1152 0.0072
ndisease	0.0346 0.0024	0.0388 0.0024	0.0409 0.0023		0.0405 0.0023
female	0.1585 0.0334	0.1667 0.0346	0.2084 0.0310		0.2054 0.0310
aveage					0.0134
avelfam					0.0093 -0.2848
avechild					0.1202 -0.0545
_cons	0.7765 0.0717	0.7574 0.0831	0.2873 0.0829		0.3807 0.0778
/lnalpha		0.0251 0.0905			
/lnsig2u			0.0550 0.0271		0.0550 0.0270
Statistics					
N	20186	20186	20186	17791	20186
ll		-43241	-43227	-24173	-43210

Legend: b/se

The PA and RE parameter estimates are quite similar; the alternative RE estimates based on normally distributed random effects are roughly comparable, whereas the FE and CRE estimates for the time-varying regressors are quite different.

22.6.8 Panel probit prediction and MEs

The `predict` postestimation command has several options that vary depending on whether the `xtpoisson` command was used with the `pa`, `re`, or `fe` option.

After the `xtpoisson, pa` command, the default `predict` option is `mu`, which gives the predicted conditional mean. After the `xtpoisson, re` command, the default prediction option `xb` gives $\mathbf{x}'_{it}\hat{\boldsymbol{\beta}}$; option `expression(exp(predict(xb)))` used with `margins` gives $\exp(\mathbf{x}'_{it}\hat{\boldsymbol{\beta}})$. After the `xtpoisson, re` command, the default option `n` estimates $E(y_{it}|\mathbf{x}_{it}) = \exp(\mathbf{x}'_{it}\hat{\boldsymbol{\beta}})$, which requires numerically integrating out α_i . After the `xtpoisson, fe` command, the default prediction option `xb` gives $\mathbf{x}'_{it}\hat{\boldsymbol{\beta}}$, while the default option `nu0` gives $\exp(\mathbf{x}'_{it}\hat{\boldsymbol{\beta}})$ by setting $\alpha_i = 1$ or equivalently $\gamma_i = \ln \alpha_i = 0$.

The default options for the `margins` command can differ in some cases from those for the `predict` command. The following commands explicitly state what option is being used in obtaining the AMEs for various panel Poisson models.

```

. * Compute AMEs for PA, RE, FE, and CRE models
. qui xtpoisson mdu $xlist, pa corr(unstr) vce(robust)
. qui margins, dydx(*) predict(mu)
. matrix PA = r(b)
. qui xtpoisson mdu $xlist, re vce(robust)
. qui margins, dydx(*) expression(exp(predict(xb)))
. matrix RE = r(b)
. qui xtpoisson mdu $xlist, re normal vce(robust)
. qui margins, dydx(*) predict(n)
. matrix REnorm = r(b)
. qui xtpoisson mdu $xlist, fe vce(robust)
. qui margins, dydx(*) predict(nu0)
. matrix FE_0 = r(b)
. qui xtpoisson mdu $xlist aveage avelfam avechild, re normal vce(robust)
. qui margins, dydx(*) predict(n)
. matrix CREall = r(b)
. matrix CRE = CREall[1..1, 1..6]
. qui margins, dydx(*) predict(nu0)
. matrix CRE_Oall = r(b)
. matrix CRE_O = CRE_Oall[1..1, 1..6]

```

The AMEs are as follows.

```

. * Report AMEs for PA, RE, FE, and CRE models
. matrix rowjoinbyname ME = PA RE REnorm FE_0 CRE CRE_O
. matrix rownames ME = PA RE REnorm FE_0 CRE CRE_O
. matrix list ME, format(%8.4f)

```

	age	lfam	child	lcoins	ndisease	female
PA	0.0089	-0.4067	0.2931	-0.2326	0.1001	0.4584
RE	0.0056	-0.3924	0.3143	-0.2549	0.1125	0.4839
REnorm	0.0082	-0.4442	0.2269	-0.3525	0.1258	0.6415
FE_0	-0.0100	0.0784	0.0948	.	.	.
CRE	-0.0346	0.2692	0.3267	-0.3556	0.1250	0.6338
CRE_O	-0.0204	0.1587	0.1926	-0.2097	0.0737	0.3737

The AMEs for PA and RE estimators are roughly comparable. The AMEs for the FE estimator are quite different. The CRE AMEs have the same sign as the FE AMEs but differ quite substantially in magnitude. The second set of CRE AMEs still differ from the FE AMEs even though both evaluate at $\gamma_i = \ln \alpha_i = 0$.

22.6.9 Negative binomial estimators

The preceding analysis for the Poisson can be replicated for the negative binomial. The negative binomial has the attraction that, unlike Poisson, the estimator is designed to explicitly handle overdispersion, and count data are usually overdispersed. This may lead to improved efficiency in estimation and a default estimate of the VCE that should be much closer to the cluster-robust estimate of the VCE, unlike for Poisson panel commands. At the same time, the Poisson panel estimators rely on weaker distributional assumptions—essentially, correct specification of the mean—and it may be more robust to use the Poisson panel estimators with cluster-robust standard errors.

For the pooled negative binomial, the issues are similar to those for pooled Poisson. For the pooled negative binomial, we use the `nbreg` command with the `vce(cluster id)` option. For the PA negative binomial, we can use the `xtnbreg` command with the `pa` and `vce(robust)` options.

For the panel negative binomial RE, we use `xtnbreg` with the `re` option. The negative binomial RE model introduces two parameters in addition to β that accommodate both overdispersion and within correlation. An estimator called the negative binomial FE estimator is no longer viewed as being a true FE estimator because, for example, it is possible to estimate the coefficients of time-invariant regressors in addition to time-varying regressors. A more complete presentation is given in, for example, [Cameron and Trivedi \(2013, chap. 9\)](#) and in [XT] **xtnbreg**.

We apply the Poisson PA and negative binomial PA and RE estimators to the doctor-visits data. We have

```
. * Comparison of negative binomial panel estimators
. qui xtpoisson mdu lcoins ndisease female age lfam child, pa
>     corr(exch) vce(robust)
. estimates store PPA_ROB
. qui xtnbreg mdu lcoins ndisease female age lfam child, pa
>     corr(exch) vce(robust)
. estimates store NBPA_ROB
. qui xtnbreg mdu lcoins ndisease female age lfam child, re
. estimates store NBRE
```

```
. estimates table PPA_ROB NBPA_ROB NBRE, eq(1) b(%8.4f) se
> stats(N ll) stfmt(%8.0f)
```

Variable	PPA_ROB	NBPA_ROB	NBRE
#1			
lcoins	-0.0815 0.0079	-0.0865 0.0078	-0.1073 0.0062
ndisease	0.0347 0.0024	0.0376 0.0023	0.0334 0.0020
female	0.1609 0.0338	0.1649 0.0343	0.2039 0.0263
age	0.0032 0.0016	0.0026 0.0016	0.0023 0.0012
lfam	-0.1487 0.0299	-0.1633 0.0291	-0.1434 0.0251
child	0.1121 0.0444	0.1154 0.0452	0.1145 0.0385
_cons	0.7755 0.0724	0.7809 0.0730	0.8821 0.0663
/ln_r			1.1280 0.0269
/ln_s			0.7259 0.0313
Statistics			
N	20186	20186	20186
ll			-40661

Legend: b/se

The Poisson and negative binomial PA estimates and their standard errors are similar. The RE estimates differ more and are closer to the Poisson RE estimates given in section [22.6.4](#).

22.7 Panel quantile regression

We covered linear conditional QR analysis of cross-sectional data in chapter 15. Pooled conditional QR is straightforward. Extension to panel data with individual-specific effects remains in developmental stages. A standard method for controlling for unobserved individual specific heterogeneity is by introducing an additive random effect with known distribution and then integrating this out. This approach will not work, because QR is a distribution-free estimator. Instead, we present FE estimation.

22.7.1 Pooled QR estimator

Pooling or population averaging introduces clustering as an additional new element to be accounted for in obtaining robust variance estimates.

Section 15.3.6 presented the community-contributed `qreg2` command, a wrapper for `qreg` that generates standard errors and t statistics that are heteroskedastic robust (the default) or cluster–robust (option `cluster()`). Another recently proposed method of robust variance estimation is the wild gradient bootstrap by [Hagemann \(2017\)](#), but we are not aware of any supporting Stata software.

We begin with the standard QR analysis, treating the sample as a pooled sample of cross-sectional observations and obtaining cluster–robust standard errors. Application is to the panel dataset on log wage used in chapter 8 with $N = 595$ and $T = 7$. For all calculations, we chose $q = .25$ and $q = .75$.

```
. * Pooled data quantile regression ignoring individual effects
. qui use mus208psid, clear
. qui qreg2 lwage exp exp2 wks ed, quant(.25) cluster(id)
. estimates store Pool25
. qui qreg2 lwage exp exp2 wks ed, quant(.75) cluster(id)
. estimates store Pool75
```

The estimates will be compared below with FE estimates. From output not given, the cluster–robust standard errors are 50%–100% larger than heteroskedastic–robust standard errors.

22.7.2 Panel QR with fixed effects

Several different approaches to conditional QR with FE models have been proposed; [Canay \(2011\)](#) provides a brief summary and discusses the limitations of these approaches. He proposes a simple estimator in the special case that $T \rightarrow \infty$, in addition to our standard assumption that $N \rightarrow \infty$. While requiring $T \rightarrow \infty$ is a limitation, he presents Monte Carlo evidence that the method works well for $T = 20$ and possibly lower.

The method assumes a linear regression with an additive separable individual effect (α_i) and an additive independent and identically distributed random error ε_{it} . We estimate the linear regression of y_{it} on vector \mathbf{x}_{it} and derive the regression residuals, which are treated as an estimate of $(\alpha_i + \varepsilon_{it})$. The individual specific average of the residuals is a consistent estimate of the fixed effects, denoted $\hat{\alpha}_i$, provided $T \rightarrow \infty$. This average is subtracted from the dependent variable, and the QR analysis is applied to the transformed variable.

The two-step estimation method is simple to implement. More difficult is the computation of standard errors because the sandwich term in the variance matrix is the sum of four terms. [Canay \(2011\)](#) conjectures that one can apply the bootstrap pairs method to the two-step estimator (see section 12.4.5) and provides supporting Monte Carlo evidence.

For illustration, we apply the method to the log wage data, even though $T = 7$ is quite small. We first create the transformed dependent variable.

```
. * FE QR: (1) Filter out the individual FE from the y variable
. qui regress lwage exp exp2 wks ed
. predict residuals_i, resid
. sort id
. by id: egen alphahat_i=mean(residuals_i)
. summarize alphahat_i
Variable |       Obs        Mean      Std. dev.       Min       Max
alphahat_i |    4,165    1.94e-10     .3257519   -.9955915    .9009727
. gen lwagehat=lwage-alphahat_i
```

We then obtain the second-stage estimates. The associated standard errors are obtained by a simpler bootstrap that bootstraps only the second stage of the estimation procedure. This does not adjust for the imprecision due to first-stage estimation of the fixed effects; the correct bootstrap should bootstrap both stages.

```
. * FE QR: (2) QR of new y with bootstrap covariance estimates
. set seed 10101
. qui bsqreg lwagehat exp exp2 wks ed, quant(.25) reps(400)
. estimates store FE25boot
. qui bsqreg lwagehat exp exp2 wks ed, quant(.75) reps(400)
. estimates store FE75boot
. xtset id

Panel variable: id (balanced)
. qui bootstrap, reps(400) seed(10101) cluster(id):
>     qreg lwagehat exp exp2 wks ed, quant(.25)
. estimates store FE25clu
. qui bootstrap, reps(400) seed(10101) cluster(id):
>     qreg lwagehat exp exp2 wks ed, quant(.75)
. estimates store FE75clu
```

Finally, all results are collected and tabulated for comparison.

```
. * Compare OLS, FE, Pooled_QR (0.22., 0.75), and Panel_QR(0.25, 0.75)
. estimates table Pool25 FE25boot FE25clu Pool75 FE75boot FE75clu, b(%8.4f) se
```

Variable	Pool25	FE25boot	FE25clu	Pool75	FE75boot	FE75clu
exp	0.0488 0.0067	0.0498 0.0016	0.0498 0.0015	0.0406 0.0059	0.0551 0.0018	0.0551 0.0019
exp2	-0.0008 0.0002	-0.0008 0.0000	-0.0008 0.0000	-0.0006 0.0001	-0.0008 0.0000	-0.0008 0.0000
wks	0.0049 0.0023	0.0031 0.0009	0.0031 0.0009	0.0068 0.0032	0.0027 0.0008	0.0027 0.0009
ed	0.0822 0.0055	0.0796 0.0017	0.0796 0.0015	0.0737 0.0063	0.0807 0.0016	0.0807 0.0015
_cons	4.5858 0.1521	4.7977 0.0578	4.7977 0.0550	5.1834 0.1932	4.9962 0.0483	4.9962 0.0488

Legend: b/se

Broadly summarized, the results indicate that FE adjustment changed the pooled estimation results by not much, aside from the coefficient of `ed`. The

standard errors from panel FE estimations are considerably smaller than those from pooled estimation. A similar large reduction in the standard errors was found in section 8.8.3 for linear regression using the same data, so this is not solely due to our failure here to allow for first-stage estimation of the fixed effects. The clustered and nonclustered bootstraps yielded similar standard errors. Panel conditional QR models with fixed effects cause a problem because inclusion of individual fixed effects changes the interpretation of the treatment-effect coefficient. The community-contributed `qregpd` command, developed by [Baker \(2016\)](#), is one option.

22.8 Endogenous regressors in nonlinear panel models

The combination of nonlinearity and endogenous regressors in panel models poses a difficult specification and estimation problem for fully parametric models. Thus, methods typically place some restrictions on the model. For example, the model structure may be restricted to permit only recursive dependence (see section 7.2.3); only RE models might be considered; only restricted forms of unobserved heterogeneity may be admitted; and so forth.

RE models that allow for endogeneity and sample selection can be estimated using the extended regression model commands `xteregress`, `xteprobit`, `xteoprobit`, and `xtintreg`; see section [23.7](#). This approach is restricted to recursive models with normal errors.

Semiparametric methods are easier to use. Specifically, in section [20.7.3](#), we presented IV methods for handling endogenous regressors in a count-data model with cross-sectional data. The same approach can be used also for panel data. That is, we treat panels as pooled cross-sections and apply nonlinear generalized method of moments (GMM) (or instrumental variables) based on a moment condition. Given valid instruments, the estimator is consistent, but such a solution is generally not efficient, because it often ignores salient features of the data. The [R] `gmm` manual entry provides code for Poisson panel regression with endogenous regressors; see also [Cameron and Trivedi \(2013\)](#), chap. 9). This nonlinear GMM approach can be applied to other cases, such as panel logit or probit with endogenous regressors.

Alternative model-specific methods accommodate complications such as the outcome variable is a count, possibly with excess zeros; an endogenous regressor may be binary or count with excess zeros; counted outcomes may be conditionally correlated or clustered; or unobserved heterogeneity is not explicitly acknowledged. Such data features are central to panel models, and hence ignoring them is unsatisfactory. Furthermore, it is particularly important to robustify standard error estimates.

22.9 Additional resources

The Stata panel commands cover the most commonly used panel methods, especially for short panels. This topic is exceptionally vast, and there are many other methods that provide less-used alternatives to the methods covered in Stata as well as methods to handle complications not covered in Stata, especially the joint occurrence of several complications such as a dynamic FE logit model. Many of these methods are covered in the panel-data books by [Arellano \(2003\)](#), [Baltagi \(2021\)](#), [Hsiao \(2014\)](#), and [Lee \(2002\)](#); see also [Rabe-Hesketh and Skrondal \(2022\)](#) for the mixed-model approach and [Cameron and Trivedi \(2013\)](#), chap. 9) for count-data models. [Cameron and Trivedi \(2005\)](#) and [Wooldridge \(2010\)](#) also cover some of these methods.

22.10 Exercises

1. Consider the panel logit estimation of section [22.4](#). Compare the following three sets of estimated standard errors for the pooled logit estimator: default, heteroskedasticity robust, and cluster–robust. How important is it to control for heteroskedasticity and clustering? Show that the `pa` option of the `xtlogit` command yields the same estimates as the `xtgee` command with the `family(binomial)`, `link(logit)`, and `corr(exchangeable)` options. Compare the PA estimators with the `corr(exchangeable)`, `corr(ar2)`, and `corr(unstructured)` options, in each case using the `vce(robust)` option.
2. Consider the panel logit estimation of section [22.4](#). Drop observations with `id > 125200`. Estimate the parameters of the FE logit model by using `xtlogit` as in section [22.4](#). Then, estimate the parameters of the same model by using `logit` with DVs for each individual (so use `xi: logit` with regressors, including `i.id`). This method is known to give inconsistent parameter estimates in a short panel. Compare the estimates with those from command `xtlogit`. Are the same parameters identified?
3. For the parameters of the panel logit models in section [22.4](#), estimate by using `xtlogit` with the `pa`, `re`, and `fe` options. Compute the following predictions: for `pa`, use `predict` with the `mu` option; for `re`, use `predict` with the `pu0` option; for `pa`, use `predict` with the `pu0` option. For these predictions and for the original dependent variable, `dmdu`, compare the sample average value and the sample correlations. Then, use the `margins`, `dydx()` command with these predict options, and compare the resulting MES for the `lcoins` variable.
4. Generate a panel dataset using the commands below. Explain why the sample is generated by the probit RE model with normally distributed errors. Obtain the probit RE estimates. Are the coefficient estimates what you expect? Explain. Next, obtain the probit PA estimates. Do you obtain similar parameter estimates? Explain. Next, obtain the predicted probabilities from the two models, and OLS regress one prediction on the other. Comment. Then, obtain the MES at $x_1 = 1$ for the two models and compare. Comment. Provide a summary comparing the PA

model with the RE model. Finally, instead repeat, except generate $e = (\text{rchi2}(4) - 4) / \sqrt{2^4}$. Comment on any major changes.

```
* Generate panel data sample for probit regression
clear all
set obs 10000
set seed 10101
generate id = _n
generate double alpha = rnormal(0,1)
generate double z1 = rnormal(0,1)
generate double z2 = z1 + rnormal(0,1)
expand 3
sort id
by id: gen t = _n
generate double x1 = z1 + rnormal(0,1)
generate double x2 = z2 + rnormal(0,1)
generate double xb = 1 + 1*x1 + 1*x2
generate double e = rnormal(0,1)
generate double u = e + alpha
generate double y = xb + u > 0
```

5. For the panel tobit model in section [22.5](#), compare the results from `xttobit` with those from `tobit`. Which do you prefer? Why?
6. Consider the panel Poisson estimation of section [22.6](#). Compare the following three sets of estimated standard errors for the pooled Poisson estimator: default, heteroskedasticity robust, and cluster-robust. How important is it to control for heteroskedasticity and clustering?
Compare the PA estimators with the `corr(exchangeable)`, `corr(ar2)`, and `corr(unstructured)` options, in each case using both the default estimate of the VCE and the `vce(robust)` option.
7. Consider the panel count estimation of section [22.6](#). To reduce computation time, use the `drop if id > 127209` command to use 10% of the original sample. Compare the standard errors obtained by using default standard errors with those obtained by using the `vce(bootstrap)` option for the following estimators: Poisson RE, Poisson FE, negative binomial RE, and negative binomial FE. How important is it to use panel-robust standard errors for these estimators?
8. Repeat the analysis of section [22.7.2](#) with the following changes: 1) set $q = 0.4$; 2) use the default (not bootstrap) standard errors. Compare the results for `qreg`, `qreg2`, and Canay's FE estimator.

Chapter 23

Parametric models for heterogeneity and endogeneity

23.1 Introduction

Two topics—unobserved heterogeneity and endogeneity—have featured throughout the book, both individually and in combination. Their treatment was for the most part in the context of subject matter dealing with specific models, such as binary outcome, tobit, and event count models.

In this chapter, we deal with these topics from a command-based perspective in which the unifying feature is the underlying assumptions and computational methodology used to handle these complications that are common across very wide classes of models.

We cover the `fmm` prefix, various `me` commands, the `sem` and `gsem` commands, and the commands for extended regression models (ERMS). These commands implement methods that are generally fully parametric and in many cases are based on joint normally distributed model errors.

The `fmm` (finite mixture model) prefix and `me` (mixed-effects) commands provide widely used methodologies for models with unobserved heterogeneity. The prefix `fmm` was introduced and applied in sections 14.2 and [20.5](#) and uses a discrete distribution for unobserved heterogeneity. The `me` commands are for various nonlinear models that use the continuous normal distribution to model unobserved heterogeneity; the related command `mixed` was presented for linear models in section 6.7.

The `SEM` (structural equation model) commands cover a wide range of systems of equations that are linear in fully observed variables and latent variables. Examples include seemingly unrelated regressions (SUR), simultaneous equations, factor analysis, and linear mixed models. These models can include endogenous regressors.

The commands for ERMS cover continuous, binary, ordered discrete, and interval outcomes. Options allow for endogeneity and binary sample selection or tobit sample selection, topics jointly considered in section [17.9](#). Additional options of the `ERM` commands implement some of the treatment-effects methods presented in chapters [24](#) and [25](#).

In the remainder of this chapter, we shall deal sequentially with the aforementioned commands, giving relatively more space to applications and features of models that have not been covered earlier in the book. Given the similarity of the syntax for different models, and given also the similarity of the motivation underlying the methodology, we keep the discussion brief, especially in those cases where a more detailed coverage has been given previously for selected models, such as for linear regression and count regression.

23.2 Finite mixtures and unobserved heterogeneity

Unobserved heterogeneity is pervasive in microeconometric models. It calls for special attention in panel-data regressions, regression with within-cluster correlation, survival data models, and treatment evaluation.

A common representation of unobserved heterogeneity is a random-effects model in which the model error includes a panel-specific or cluster-specific additive independent and identically distributed (i.i.d.) random variable that is uncorrelated with other random variables. Both parametric [for example, maximum likelihood (ML)] and semiparametric methods (for example, generalized least squares) for handling these cases have appeared in various sections of this book.

A richer representation of unobserved heterogeneity postulates randomness in parameters. This representation is neither additive nor separable. The heterogeneity may be represented by either a continuous or a discrete distribution. The `mixed`, `sem`, `gsem`, and `ERM` commands usually characterize randomness by a continuous distribution, often the normal distribution; an exception is the `lclass()` option of the `gsem` command, which uses a discrete distribution. The `fmm` prefix for finite mixture models always characterizes parameter heterogeneity using a discrete distribution.

23.2.1 Finite mixtures model

A parametric finite mixture model (FMM) is constructed by taking a convex combination of a finite number of distributions (components) with different parameters. A mixture distribution does not belong to the same class as the component distribution; for example, a mixture of normals is not normal. In practice, one does not know either the mixing parameters or which component distribution any particular observation comes from. Hence, the components are often referred to as latent classes, and the FMM is also known as a latent class model.

There are several motivations for using the FMM framework. It has many more parameters relative to the benchmark model with a single distribution and hence has greater flexibility. It can be interpreted as a way of generating good

approximations to unknown distributions. In specific cases, it can support insightful interpretations of differences in subpopulations.

A parametric finite mixture regression model with C components is written as

$$f(y_i | \mathbf{x}_i, \mathbf{z}_i) = \sum_{j=1}^C \pi_j(\mathbf{z}_{ij}) f_j(y_i | \mathbf{x}_{ij}; \boldsymbol{\theta}_j), \quad i = 1, \dots, N; \quad j = 1, \dots, C \quad (23.1)$$

where for the j th component the usual regressors are denoted as \mathbf{x}_j , $\boldsymbol{\theta}_j$ refers to the unknown j -class parameters, and the class probability π_j is a function of observable variables \mathbf{z}_{ij} that determines the weight of the j th component. The number of mixture components, C , is usually unknown and is treated as an additional parameter.

The component probabilities should satisfy $0 < \pi_{ij} \leq 1$ and $\sum_{j=1}^C \pi_{ij} = 1$. To ensure that these constraints are satisfied, one should further parameterize the π functions as a multinomial logit model, with

$$\pi_j(\mathbf{z}_{ij}) = \frac{\exp(\mathbf{z}'_{ij} \boldsymbol{\gamma}_j)}{\sum_{l=1}^C \{1 + \exp(\mathbf{z}'_{il} \boldsymbol{\gamma}_l)\}}$$

This simplifies to a logit model in the common case that $C = 2$. Often, a simpler FMM with constant class probabilities is estimated; then $\pi_j(\mathbf{z}_{ij}) = \pi_j$.

In this specification, variation in \mathbf{x}_j has an impact on the outcome that varies depending upon the latent class. Moreover, component-specific restrictions can be imposed. For example, exclusion restrictions on either or both \mathbf{x} variables or \mathbf{z} variables may vary across the C components.

FMMs account for between-class heterogeneity in responses. A more general model additionally allows for unobserved heterogeneity within each component. Then the distribution of the component-specific outcome y_{ij} depends upon both exogenous variables \mathbf{x}_{ij} and on some individual-specific continuously distributed unobserved factor, say, ξ_i . For example, we may specify

$$f(y_i | \mathbf{x}_i, \mathbf{z}_i, \xi_i) = \sum_{j=1}^C \pi_j(\mathbf{z}_{ij}) f_j(y_{ij} | \mathbf{x}_{ij}, \xi_i; \boldsymbol{\theta}_j)$$

In this richer specification, there are now two types of unobserved heterogeneity, one discrete and the other continuous. The standard assumption is that the two types are independent of each other and independent of the regressors; hence, the model is a random-effect-type specification. An example is the mixture of negative binomials studied in section 20.5.5; formally, this is a discrete two-component mixture of negative binomials, each of which can be interpreted as a (continuous) Poisson-gamma mixture.

Adding an additional component can considerably increase the number of parameters estimated. Models with a number of components that lead to the lowest value of the Akaike or Bayes information criterion (AIC and BIC), defined in section 13.8.2, are preferred. BIC has a bigger penalty for additional parameters. These criteria are used rather than a formal statistical test of whether to add an additional component.

23.2.2 The `fmm` prefix

The syntax of the `fmm` prefix is detailed in section 14.2.5.

For convenience, we repeat the generic syntax,

```
fmm #: model depvar [indepvars] [, options]
```

where # refers to the number of components in the specification and *model* is an estimation command that will vary with the specified distribution.

The syntax for the more general hybrid version of the model that allows parameterization of the latent class probabilities is as follows,

```
fmm #: model depvar indepvars[, lcprob(varlist) options]
```

where *varlist* refers to the variables that are used to parameterize the latent class probabilities.

An important option is the `vce()` option. FMMS are estimated by ML under the assumption of completely correct model specification. For some of these models, estimators are inconsistent if the i.i.d. assumption fails, even if other aspects of the model are correctly specified. We nonetheless report robust standard errors in such cases because these still give a better estimate of estimator precision; see section 13.3.1. Because FMMS are more flexible models and incorporate heteroskedasticity, we expect them to reduce any difference between default and heteroskedastic–robust standard errors.

A detailed FMM application for the linear regression model under normality is presented in section 14.3, and a count application is given in section [20.5](#). The command can be applied to many other commonly used parametric model commands, such as `fmm: logit` for a logit model. The command covers censored, truncated, and interval linear regression (`tobit`, `truncreg`, `intreg`); instrumental variables (`ivregress`); binary outcomes (`logit`, `probit`, `cloglog`); multinomial outcomes (`mlogit`, `ologit`, `oprobit`); counts (`poisson`, `nbreg`, `tpoisson`); durations (`streg`); beta regression (`betareg`); and generalized linear models (`glm`).

There is a unified and robust computational algorithm for fitting these models by an ML algorithm that is a two-part hybrid consisting of the expectation–maximization algorithm in the first part and a gradient-based Newton–Raphson-type algorithm in the second. Convergence of the algorithm is not guaranteed in general and is likely to be case dependent. Models with more components, and hence more parameters, are more challenging. Models with many components, including one or more with a small subpopulation, will be harder to identify.

23.3 Empirical examples of FMMs

The use of the finite mixture framework helps modeling and interpretation of heterogeneous responses in several ways. First, we can test whether the latent class framework leads to an improvement in the fit of the model. Second, we can make comparisons across latent classes between the outcomes' responses, as reflected in coefficients, marginal effects (ME), and so forth, resulting from changes in the regressors. Third, if the latent classes can be identified by their observable characteristics, then interpretation of the results potentially can be more informative in the sense that each latent class refers to a specific subpopulation.

We present a series of nonlinear regression examples that may be viewed as FMM extensions of FMM examples already used in sections 14.2–14.3 and [20.5](#). The first example is presented in greater detail than the remaining examples.

23.3.1 Example 1: Gamma regression for expenditures

The distribution of medical expenditures is well known to be thick tailed and right skewed and to have significant nonnormal kurtosis, features that pose difficulties in both modeling and forecasting. One well-established empirical approach is to apply a log transformation to positive expenditures and then assume a normal distribution. Another approach is to not apply the transformation and instead assume that the conditional distribution is gamma. By better accounting for large expenditures, the gamma model may provide a better fit to the data than the log-normal, and the gamma model has the advantage of directly modeling the level of expenditure.

We use the sample data on healthcare expenditures of the Medicare elderly first used in section 3.2 and estimate a two-component finite mixture of gamma regression models for the level of medical expenditures. Gamma regression requires positive values for the dependent variable, so we drop the 3% of the sample with 0 expenditures.

```
. * Gamma: Read chapter 3 example data and truncate expenditure at zero
. qui use mus203mepsmedexp
. keep if totexp > 0
(109 observations deleted)
. describe totexp totchr age female educyr private hvgg
```

Variable name	Storage type	Display format	Value label	Variable label
totexp	double	%12.0g		Total medical expenditure
totchr	double	%12.0g		# of chronic problems
age	double	%12.0g		Age
female	double	%12.0g	female	Female
educyr	double	%12.0g		Years of education
private	double	%12.0g	private	Private supplementary insurance
hvgg	float	%9.0g	hvgg	Health status is excellent, good, or very good

The gamma distribution is a member of the generalized linear model (GLM) family (see section 13.3.7) and can be estimated using the `glm` command with options `family(gamma)` and `link(log)`; the latter option specifies an exponential conditional mean function. We obtain

```
. * Gamma: Standard gamma regression (one component) as benchmark
. glm totexp totchr age female educyr private hvgg, family(gamma) link(log)
> vce(robust) nolog

Generalized linear models
Optimization : ML
Number of obs = 2,955
Residual df = 2,948
Scale parameter = 2.660609
Deviance = 4236.116216
(1/df) Deviance = 1.436946
Pearson = 7843.476403
(1/df) Pearson = 2.660609
Variance function: V(u) = u^2
[Gamma]
Link function : g(u) = ln(u)
[Log]
AIC = 19.55801
BIC = -19322.1
Log pseudolikelihood = -28889.96311
```

totexp	Coefficient	Robust				
		std. err.	z	P> z	[95% conf. interval]	
totchr	.3418083	.0223432	15.30	0.000	.2980165	.3856001
age	.0056696	.0049044	1.16	0.248	-.0039428	.015282
female	-.1124211	.0602113	-1.87	0.062	-.2304332	.0055909
educyr	.0272675	.0099605	2.74	0.006	.0077452	.0467897
private	.1062875	.0638005	1.67	0.096	-.0187591	.2313342
hvgg	-.3319719	.0643275	-5.16	0.000	-.4580515	-.2058923
_cons	7.617334	.3996511	19.06	0.000	6.834033	8.400636

Two-component model parameter estimates

The two-component gamma mixture model estimates are

```
. * Gamma: Two-component finite mixture gamma regression
. fmm 2, nolog vce(robust): glm totexp totchr age female educyr private hvgg,
>      family(gamma) link(log)
Finite mixture model                                         Number of obs = 2,955
Log pseudolikelihood = -28554.52
```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1.Class	(base outcome)					
2.Class						
_cons	-1.105388	.1766837	-6.26	0.000	-1.451682	-.7590941

Class: 1
 Response: totexp
 Model: glm, family(gamma)

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
totexp						
totchr	.4209618	.0212731	19.79	0.000	.3792672	.4626564
age	.0212328	.0041791	5.08	0.000	.0130418	.0294237
female	-.0531238	.0518319	-1.02	0.305	-.1547125	.0484649
educyr	.0399077	.0080321	4.97	0.000	.0241651	.0556503
private	.2649818	.0561469	4.72	0.000	.154936	.3750277
hvgg	-.0969274	.0570488	-1.70	0.089	-.2087409	.0148862
_cons	5.229722	.347071	15.07	0.000	4.549475	5.909968
/totexp						
logs	-.1713974	.0221162			-.2147444	-.1280504

Class: 2
 Response: totexp
 Model: glm, family(gamma)

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
totexp						
totchr	.2683625	.0415283	6.46	0.000	.1869684	.3497565
age	-.0077785	.0081555	-0.95	0.340	-.023763	.008206
female	-.1488827	.0904229	-1.65	0.100	-.3261083	.028343
educyr	.0136653	.0162006	0.84	0.399	-.0180873	.0454179
private	.0684348	.0970268	0.71	0.481	-.1217343	.2586039
hvgg	-.3889646	.0983737	-3.95	0.000	-.5817735	-.1961556
_cons	9.873871	.7002255	14.10	0.000	8.501454	11.24629
/totexp						
logs	-.019757	.0294952			-.0775667	.0380526

Adding a second component greatly improved the fit of the model: the log likelihood increased from $-28,890$ to $-28,555$, leading to the BIC, obtained using the `estat ic` postestimation command, dropping very substantially from 57,836 to 57,245.

The output displays parameter estimates for the two latent classes. These estimates differ considerably, which indicates heterogeneity in responses between the two groups. For example, the ratio of coefficients of `totchr` in the two components is about 1.57.

Mean probabilities and mean expenditures in each class

The `estat lcprob` postestimation command generates the estimated population proportions of the two latent classes. These are approximately 0.75 and 0.25, respectively. The `estat lcmean` command generates the class mean expenditures.

```
. * Gamma: Latent class marginal probabilities and latent class distribution means
. estat lcprob
```

Latent class marginal probabilities	Number of obs = 2,955
-------------------------------------	-----------------------

	Delta-method			
	Margin	std. err.	[95% conf. interval]	
Class				
1	.7512683	.0330159	.681157	.8102571
2	.2487317	.0330159	.1897429	.318843

```
. estat lcmean
```

Latent class marginal means	Number of obs = 2,955
-----------------------------	-----------------------

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
1						
totexp	4033.326	204.0074	19.77	0.000	3633.479	4433.174
2						
totexp	17254.26	1383.756	12.47	0.000	14542.15	19966.37

The larger class has a lower mean medical expenditure than the smaller class, about \$4,033 versus \$17,254. So about 75% of the observations are drawn from a low-mean distribution and the rest come from a high-mean distribution.

MEs

Applying the default form of the `margins, dydx()` command provides a weighted average of the latent class-specific average marginal effect (AME), using the class probability as weight.

```

. * Gamma: ME of change in totchr in one- and two-component models
. margins, dydx(totchr) noatlegend // AME in two-component model
Average marginal effects                                         Number of obs = 2,955
Model VCE: Robust
Expression: Predicted mean (Total medical expenditure), using class
probabilities, predict(mu outcome(totexp))
dy/dx wrt: totchr

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
totchr	2427.287	206.3907	11.76	0.000	2022.769	2831.805

```

. qui glm totexp totchr age female educyr private hvgg, family(gamma)
> link(log) vce(robust)

. margins, dydx(totchr) noatlegend // AME in one-component model
Average marginal effects                                         Number of obs = 2,955
Model VCE: Robust
Expression: Predicted mean totexp, predict()
dy/dx wrt: totchr

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
totchr	2530.212	197.6586	12.80	0.000	2142.809	2917.616

The AME of an additional chronic condition (totchr) is \$2,427 in the two-component model, similar to \$2,530 from a standard (one-component) gamma regression.

This example illustrates that if the target parameter is the weighted average of AMEs, then a one-component model could be sufficient. This conclusion was also reached in the case of finite mixture of linear regression; see sections 14.3.5 and 14.3.9. On the other hand, if the objective is to study in detail the differences between the two latent subpopulations, then the FMM framework is informative.

Fitted means for each observation in each latent class

The `predict` command generates for each individual the predicted mean of expenditures within each latent class. The average of these predicted means

equals the mean expenditure generated by the preceding `estat lcmean` command.

We label the individual predictions `mu1` and `mu2`. By generating histograms of `mu1` and `mu2`, we can check the overlap between the two distributions. A small overlap means that the distributions are well separated and identification of latent classes is more robust.

```
. * Gamma: Generate and graph latent class predicted expenditures
. qui fmm 2, vce(robust): glm totexp totchr age female educyr private hvgg,
>      family(gamma) link(log)
. estimates store FMM2
. predict mu*
(option mu assumed)
. summarize mu1 mu2
Variable |   Obs        Mean    Std. dev.       Min       Max
mu1 | 2,955    4033.326    2977.282    710.4969   29270.69
mu2 | 2,955    17254.26     8389.956    6578.288   91909.41
. qui histogram mu1
. qui histogram mu2
```

Figure 23.1 presents the resulting histograms of the fitted means `mu1` and `mu2`. The overall separation is quite large, with a relatively small overlap between the right tail of the first distribution and the left tail of the second.

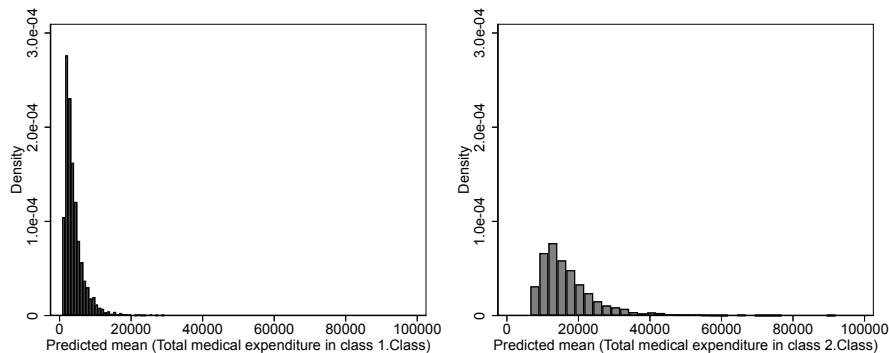


Figure 23.1. Fitted means from FMM2 gamma regression

MEs in each latent class

Given the predicted mean in each latent class, we can compute separately the AME in each class.

```
. * Gamma: Separate AME of totchr for each component
. margins, dydx(totchr) predict(mu class(1)) predict(mu class(2)) noatlegend
Average marginal effects                                         Number of obs = 2,955
Model VCE: Robust
dy/dx wrt: totchr
1._predict: Predicted mean (Total medical expenditure in class 1.Class),
             predict(mu class(1))
2._predict: Predicted mean (Total medical expenditure in class 2.Class),
             predict(mu class(2))
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
totchr _predict						
	1	1697.876	126.9435	13.38	0.000	1449.072
	2	4630.396	717.6815	6.45	0.000	3223.766
						6037.025

The AMEs of \$1,698 and \$4,630 vary greatly across the components. Note that $0.7512683 \times 1697.876 + 0.2487317 \times 4630.396 = 2427.287$, so the overall AME is the component probability weighted average of the component AMES.

Fitted posterior class probabilities

After fitting an FMM, one can estimate the posterior probability that observation i belongs to latent class j ,

$$\pi_{ij}^{\text{post}} = \frac{\pi_{ij} f_j(y_i | \mathbf{x}_{ij}, \boldsymbol{\theta}_j)}{\sum_{j=1}^C \pi_{ij} f_j(y_i | \mathbf{x}_{ij}, \boldsymbol{\theta}_j)}$$

This quantity is the contribution of the j th component to the density $f(y_i | \mathbf{x}_{ij}, \boldsymbol{\theta}_j)$ defined in (23.1). The postestimation command `predict postprob*, classpost` generates an estimate of the posterior probability of membership of every observation to each of the latent classes.

A common rule is to assign each observation to the highest probability class; every observation must belong to one of the latent classes. The mean of the posterior probabilities in each latent class is the sample estimate of the parameter π_j , the population proportion of that class. The `estat lcprob` command delivers these mean values and associated confidence intervals.

It is informative to display the distribution of posterior probabilities given by the histograms. If the overlap in the distribution of posterior probabilities is small, then the mixture components are said to be well separated.

- . * Gamma: Generate and graph latent class posterior probabilities
- . predict postprob*, classpost
- . summarize postprob*

Variable	Obs	Mean	Std. dev.	Min	Max
postprob1	2,955	.7512679	.2753035	9.27e-24	.9684005
postprob2	2,955	.2487321	.2753035	.0315995	1

- . qui histogram postprob1
- . qui histogram postprob2

Figure 23.2 shows good separation.

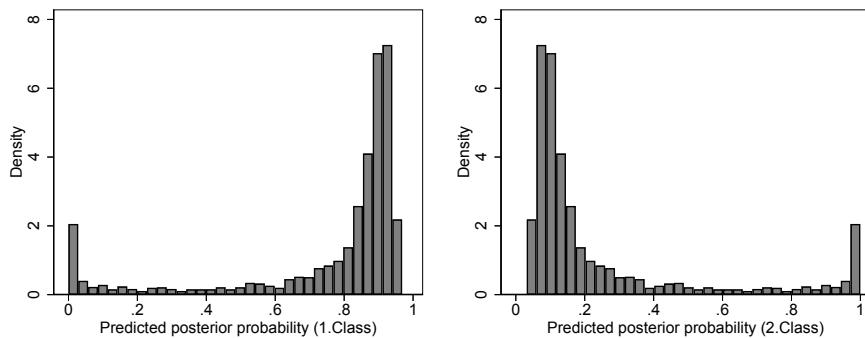


Figure 23.2. Posterior probabilities from FMM2 gamma regression

Model comparison

The two-component mixture fits the data significantly better than the one-component model. Would adding a third component further improve the fit? More generally, what is the stopping rule regarding increasing C given that

the fit of the model can always be improved, even if very slightly at times, by increasing C ?

While adding components may further improve the fit, there is also a potential for overfitting. Penalized likelihood criteria that incorporate the tradeoff between fit of the model and parsimony (number of parameters in the model) form the basis of model selection; see section 13.8.2.

Aside from fit, in applied work, interpretability of the results is also important, and in that regard, qualitative features of the fit models may also play a role in model comparison and selection.

We illustrate this approach in the context of model comparison of FMM2 and FMM3 gamma mixture specifications. Such an exercise is usually worthwhile because the “correct” number of components is unknown. However, we show only parts of the full output.

```

. * Gamma: Three-component finite mixture gamma regression
. qui fmm 3, vce(robust): glm totexp totchr age female educyr private hvgg,
>     family(gamma)
. estimates store FMM3
. estat lcprob

```

Latent class marginal probabilities Number of obs = 2,955

	Delta-method			
	Margin	std. err.	[95% conf. interval]	
Class				
1	.5082894	.0552012	.4013778	.6144482
2	.2627906	.0529629	.1725967	.378553
3	.22892	.0315554	.1729591	.296496

```
. estat lcmean
```

Latent class marginal means Number of obs = 2,955

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
1	totexp	4117.617	328.4428	12.54	0.000	3473.881 4761.353
2	totexp	4782.136	1046.427	4.57	0.000	2731.177 6833.096
3	totexp	18013.16	1508.672	11.94	0.000	15056.21 20970.1

For the FMM3 mixture, the mixing proportions are (0.51, 0.26, 0.23). The corresponding latent class means are (\$4,118, \$4,782, \$18,013), so the smallest (third) latent class also has the largest mean expenditure. The confidence intervals generated by applying the `estat lcmean` command show that the first two components have means that overlap considerably, while the third component is well separated from the other two components. The two smallest classes have significant overlap.

The `estimates stats` postestimation command gives the information criteria.

```
. * Gamma: Compare two-and three-component finite mixture gamma models
. estimates stats FMM2 FMM3
```

Akaike's information criterion and Bayesian information criterion

Model	N	ll(null)	ll(model)	df	AIC	BIC
FMM2	2,955	.	-28554.52	17	57143.04	57244.89
FMM3	2,955	.	-28514.22	26	57080.43	57236.21

Note: BIC uses N = number of observations. See [R] BIC note.

FMM3 is slightly superior to FMM2 because it has a BIC of 57,236 compared with 57,245 for the FMM2 model.

While FMM3 is a slight improvement over FMM2, the gain in terms of interpretability of results is limited given the large overlap in the distribution of predictions for components 1 and 2. Under this consideration, there is a case for preferring FMM2 over FMM3.

Allowing mixing proportions to vary

In the previous example, the mixture proportions were kept fixed. In some cases, it may be appropriate to parameterize the class probability as a function of observable variables. The resulting specification is variously known as a varying probability FMM or a “mixture-of-experts” model.

For illustration, we modify the FMM2 specification such that the class probabilities are functions of `female` and `age` and the component regressions are specified with regressors `totchr`, `age`, `female`, `educyr`, `private`, and `hvgg`. That is, `female` and `age` affect expenditures both directly and indirectly.

```
. * Gamma: Two-component model with class probabilities varying with regressors
. fmm 2, nolog lcprob(female age) vce(robust): glm totexp totchr age female
>      educyr private hvgg, family(gamma) link(log)
```

Finite mixture model Number of obs = 2,955
Log pseudolikelihood = -28551.101

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1. Class	(base outcome)					
2. Class						
female	-.1604032	.278804	-0.58	0.565	-.706849	.3860427
age	.047179	.0225356	2.09	0.036	.00301	.091348
_cons	-4.395367	1.563402	-2.81	0.005	-7.459579	-1.331155

Class: 1
Response: totexp
Model: glm, family(gamma)

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
totexp						
totchr	.4219811	.022485	18.77	0.000	.3779114	.4660507
age	.012682	.0063975	1.98	0.047	.0001432	.0252209
female	-.0079641	.0818668	-0.10	0.923	-.1684202	.1524919
educyr	.0387974	.0085443	4.54	0.000	.0220509	.0555439
private	.2706864	.058182	4.65	0.000	.1566517	.3847211
hvgg	-.0761219	.0600894	-1.27	0.205	-.1938949	.0416511
_cons	5.803233	.4794058	12.11	0.000	4.863615	6.742851
/totexp						
logs	-.1812314	.0241721			-.2286078	-.133855

```

Class:      2
Response:  totexp
Model:     glm, family(gamma)

```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
totexp						
totchr	.2733363	.0403532	6.77	0.000	.1942455	.3524271
age	-.0238067	.0120089	-1.98	0.047	-.0473437	-.0002697
female	-.0886044	.141627	-0.63	0.532	-.3661881	.1889794
educyr	.013494	.0166899	0.81	0.419	-.0192176	.0462055
private	.0822525	.0935969	0.88	0.380	-.1011942	.2656991
hvgg	-.3682712	.0952091	-3.87	0.000	-.5548776	-.1816647
_cons	10.96605	.9102416	12.05	0.000	9.182007	12.75009
/totexp						
logs	-.0177617	.0307591			-.0780484	.0425249

```
. estat lcmean
```

```
Latent class marginal means                                         Number of obs = 2,955
```

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
1						
totexp	3888.241	219.6352	17.70	0.000	3457.764	4318.718
2						
totexp	16781.73	1446.632	11.60	0.000	13946.38	19617.07

For these data, there is only a small increase in the log likelihood, from $-28,555$ to $-28,551$. The increase in fit is not enough to compensate for the addition of two parameters, and the BIC increases from 57,245 to 57,254. The output shows that the mixture probability has weak but statistically significant (at 5%) positive dependence on `age`, and it is not expected that the AME estimates would change significantly.

23.3.2 Example 2: Logit and probit regression

In binary outcome models, unobserved heterogeneity is often modeled as an additive random effect. A simple example is a logit model with random intercept—the random component could reflect heterogeneity in preferences. This specification can be further extended by allowing additive random

components in slope coefficients; it is feasible to estimate such a specification using the `melogit` command. The present FMM, in which unobserved heterogeneity has a discrete representation, is analogous to a model with finite number of “types” of individuals in the population, with each “type” having its own regression function.

For binary outcome data, rather than think of the differences in latent classes in terms of the realized outcome that takes only the values 0 and 1, we think of differences in propensity scores, where a propensity score is an estimate of the conditional probability of observing $y = 1$, given other regressors.

For illustration, we reconsider the binary logit example of section [17.4](#). A two-component FMM is estimated. The outcome is the health insurance status of an individual. Having private insurance is recorded as `ins` = 1. The regressors are health status, household income, years of education, and marital status.

. * Logit: Read binary outcome chapter example data				
. qui use mus217hrs, clear				
. describe ins hstatusg hhincome educyear married				
Variable name	Storage type	Display format	Value label	Variable label
ins	float	%9.0g		Person has private insurance
hstatusg	float	%9.0g		Health status good
hhincome	float	%9.0g		Total household income
educyear	double	%12.0g		Years of education
married	double	%12.0g		Married

The model estimates follow.

```
. * Logit: Two-component finite mixture logit regression and predictions
. fmm 2, nolog vce(robust): logit ins hstatusg hhincome educyear married
Finite mixture model                                         Number of obs = 3,206
Log pseudolikelihood = -1991.8357
```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1. Class	(base outcome)					
2. Class	_cons	-1.112667	.599617	-1.86	0.064	-2.287895 .0625605

```
Class: 1
Response: ins
Model: logit
```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
ins						
hstatusg	.1480639	.2305457	0.64	0.521	-.3037974	.5999251
hhincome	.0045661	.0016044	2.85	0.004	.0014215	.0077107
educyear	.1301162	.0322808	4.03	0.000	.066847	.1933853
married	.0206516	.6714559	0.03	0.975	-1.295378	1.336681
_cons	-3.022338	.4784035	-6.32	0.000	-3.959992	-2.084684

```
Class: 2
Response: ins
Model: logit
```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
ins						
hstatusg	2.94736	3.691413	0.80	0.425	-4.287676	10.1824
hhincome	-.0088856	.0083216	-1.07	0.286	-.0251957	.0074245
educyear	.6708137	.3743691	1.79	0.073	-.0629364	1.404564
married	6.088392	4.5532	1.34	0.181	-2.835717	15.0125
_cons	-10.34426	5.939966	-1.74	0.082	-21.98638	1.297856

```
. predict classpr*
(option mu assumed)
. twoway (histogram classpr1, width(.05)) (histogram classpr2,
>      fcolor(white) width(.05)), legend(off)
>      xtitle("Predicted marginal probabilities for the two classes") scale(1.2)
```

The coefficient estimates differ greatly across the two components. At the same time, the improvement in fit is modest. From output not given, the two-component model has AIC of 4005.7 and BIC of 4072.5 compared with corresponding values of 4025.7 and 4056.1 for the logit model. So the AIC criterion favors the mixture model, while the BIC criterion, which puts a relatively higher weight on parsimony, favors the simple logit model.

Figure 23.3 is a histogram of fitted probabilities that shows that the first component distribution has a lower mean and smaller dispersion relative to the second.

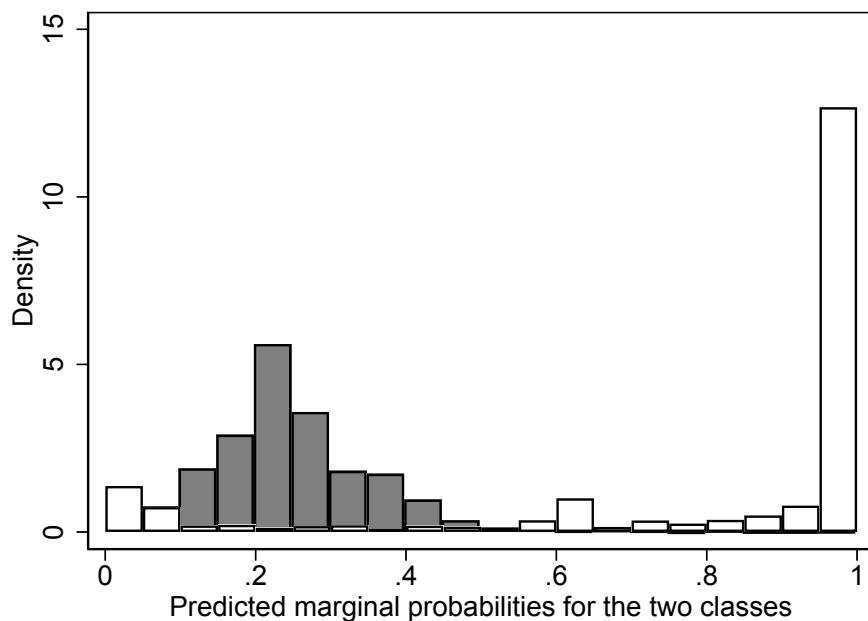


Figure 23.3. Fitted probabilities from FMM2 logit regression

The latent class means and posterior probabilities are summarized using the `estat` postestimation commands:

```
. * Logit: Latent class marginal probabilities and latent class distribution means
. estat lcmean
```

Latent class marginal means Number of obs = 3,206

		Delta-method		
	Margin	std. err.	[95% conf. interval]	
1	ins	.2540194	.055943	.1603109 .3778554
2	ins	.7979645	.0889439	.5725601 .9209218

```
. estat lcprob, classposteriorpr
```

Latent class marginal posterior probabilities Number of obs = 3,206

		Delta-method		
	Margin	std. err.	[95% conf. interval]	
Class				
1	.752626	.1111923	.4855336	.9074774
2	.247374	.1111923	.0925226	.5144664

The average fitted values of the probability of having private insurance are low (0.254) for the first latent class and high (0.798) for the second latent class. The mixture weights (or the average of the posterior probabilities for observations in each class) are 0.75 and 0.25 for class 1 and 2, respectively. These weights are used to estimate the weighted sum of the AMEs.

The computations reported above can also be reproduced for probit and complementary log–log regression by replacing `logit` in the command statement with `probit` or `cloglog`.

We next compare the AME of income on insurance status using the one-component and two-component models.

```

. * Logit: ME of change in totchr in one- and two-component models
. qui fmm 2, nolog vce(robust): logit ins hstatusg hhincome educyear married
. margins, dydx(hhincome)

Average marginal effects                                         Number of obs = 3,206
Model VCE: Robust

Expression: Predicted mean (Person has private insurance), using class
probabilities, predict(mu outcome(ins))
dy/dx wrt: hhincome

```

	Delta-method					[95% conf. interval]
	dy/dx	std. err.	z	P> z		
hhincome	.0004929	.0002403	2.05	0.040	.000022	.0009638

```

. qui logit ins hstatusg hhincome educyear married, nolog
. margins, dydx(hhincome)

Average marginal effects                                         Number of obs = 3,206
Model VCE: OIM

Expression: Pr(ins), predict()
dy/dx wrt: hhincome

```

	Delta-method					[95% conf. interval]
	dy/dx	std. err.	z	P> z		
hhincome	.0004832	.000165	2.93	0.003	.0001599	.0008065

The difference is small (0.000493 versus 0.000483), which again confirms that for estimation of the weighted AME, there is no significant gain from using the FMM framework. The main advantage of the FMM specification is that it provides information about the heterogeneous response to changes.

The parameter estimates of the model with all four regressors appearing in each component appear to be not robust. For example, the coefficients of `hstatusg` and `married` have very large standard errors. In such cases, one may wish to formally test the equality of coefficients or choose to have a different regressor for each component of the mixture. Such a model could potentially be an improvement on a one-component model.

We next illustrate how these steps are implemented. First, we generate Stata's internal name list (coefficient legend).

```

. * Logit: Obtain coefficient legend necessary for tests
. qui fmm 2, vce(robust): logit ins hstatusg hhincome educyear married
. fmm, coeflegend
Finite mixture model                                         Number of obs = 3,206
Log pseudolikelihood = -1991.8357

```

	Coefficient Legend
1.Class	(base outcome)
2.Class _cons	-1.112667 _b[2.Class:_cons]

```

Class:    1
Response: ins
Model:    logit

```

	Coefficient Legend
ins hstatusg hhincome educyear married _cons	.1480639 _b[ins:1.Class#c.hstatusg] .0045661 _b[ins:1.Class#c.hhincome] .1301162 _b[ins:1.Class#c.educyear] .0206516 _b[ins:1.Class#c.married] -3.022338 _b[ins:1.Class]

```

Class:    2
Response: ins
Model:    logit

```

	Coefficient Legend
ins hstatusg hhincome educyear married _cons	2.94736 _b[ins:2.Class#c.hstatusg] -.0088856 _b[ins:2.Class#c.hhincome] .6708137 _b[ins:2.Class#c.educyear] 6.088392 _b[ins:2.Class#c.married] -10.34426 _b[ins:2.Class]

Second, given these complex names, we apply the `test` command to test equality of coefficients.

```

. * Logit: Tests of coefficient equality and of regressor relevance
. test (_b[ins:1.Class#c.hstatusg] = _b[ins:2.Class#c.hstatusg])
>      (_b[ins:1.Class#c.hhincome] = _b[ins:2.Class#c.hhincome])
>      (_b[ins:1.Class#c.educyear] = _b[ins:2.Class#c.educyear])
>      (_b[ins:1.Class#c.married] = _b[ins:2.Class#c.married]), mtest
( 1) [ins]1bn.Class#c.hstatusg - [ins]2.Class#c.hstatusg = 0
( 2) [ins]1bn.Class#c.hhincome - [ins]2.Class#c.hhincome = 0
( 3) [ins]1bn.Class#c.educyear - [ins]2.Class#c.educyear = 0
( 4) [ins]1bn.Class#c.married - [ins]2.Class#c.married = 0

```

	chi2	df	p > chi2
(1)	0.54	1	0.4623*
(2)	2.95	1	0.0861*
(3)	2.31	1	0.1283*
(4)	2.23	1	0.1357*
All	7.40	4	0.1163

* Unadjusted p-values

A joint test does not reject the null hypothesis of equality of all coefficients at level 0.05. With individual tests, the equality hypothesis is not rejected for all variables at level 0.05. If this is viewed as a multiple testing situation (see section 11.6) and `test` command option `mtest(sidak)` is used to compute the Šidák correction, then again all variables are statistically insignificant.

A next step might be to respecify the model so class 1 regressors are `hhincome` `educyear` and class 2 regressors are `hstatusg` `educyear` `married`.

23.3.3 Example 3: Multinomial logit regression

The multinomial logit model (MNL) was considered in section [18.4](#). Together with the multinomial probit (MNP), this specification is standard for modeling unordered choices when there are more than two alternatives. MNL is computationally easier to estimate than MNP but suffers from the independence of irrelevant alternatives property that places nontrivial restrictions on the choice set; see section [18.6.1](#). One relatively straightforward way of overcoming this limitation is to replace MNL by a finite mixture that is still computationally easier to estimate than MNP.

For illustration, we use the same choice-of-fishing-mode data as used in section [18.3](#). The fishing site is one of four possible sites. For simplicity, only one regressor, `income`, is used.

```
. * MNL: Read multinomial outcome chapter example data
. qui use mus218hk, clear
. describe mode income
Variable      Storage   Display   Value
      name       type     format    label    Variable label
-----
mode          float    %9.0g      modetype  Fishing mode
income        float    %9.0g      Monthly income in thousands $
. summarize mode income
      Variable |      Obs       Mean     Std. dev.      Min      Max
                 | 1,182  3.005076  .9936162       1         4
                 | 1,182  4.099337  2.461964  .4166667      12.5
```

We fit a two-component MNL FMM.

```
. * MNL: Two-component finite mixture logit regression
. fmm 2, nolog vce(robust): mlogit mode income
```

Finite mixture model Number of obs = 1,182
Log pseudolikelihood = -1466.0383

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1.Class	(base outcome)					
2.Class						
_cons	.6965883	.1996388	3.49	0.000	.3053035	1.087873

Class: 1
Response: mode
Model: mlogit

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1.mode	(base outcome)					
2.mode						
income	.4677133	.5583008	0.84	0.402	-.6265362	1.561963
_cons	.5910911	.8454539	0.70	0.484	-1.065968	2.24815
3.mode						
income	3.737117	.941986	3.97	0.000	1.890858	5.583375
_cons	-23.02747	7.881857	-2.92	0.003	-38.47563	-7.579318
4.mode						
income	1.926167	.6855103	2.81	0.005	.5825912	3.269742
_cons	-1.985153	1.795477	-1.11	0.269	-5.504224	1.533917

Class: 2
 Response: mode
 Model: mlogit

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1.mode	(base outcome)					
2.mode						
income	-.0306097	.0889195	-0.34	0.731	-.2048887	.1436692
_cons	.0997866	.5711038	0.17	0.861	-1.019556	1.21913
3.mode						
income	.0140262	.0523073	0.27	0.789	-.0884941	.1165466
_cons	1.177968	.3004752	3.92	0.000	.5890475	1.766889
4.mode						
income	-.441771	.1714284	-2.58	0.010	-.7777645	-.1057775
_cons	1.737714	.579962	3.00	0.003	.6010095	2.874419

The coefficients in the mode equation appear to be significantly different between the two classes.

The next set of results obtains the latent class probabilities.

```
. * MNL: Latent class marginal probabilities and distribution means
. estat lcmean
```

Latent class marginal means			Number of obs = 1,182
-----------------------------	--	--	-----------------------

	Delta-method			
	Margin	std. err.	[95% conf. interval]	
1				
mode				
Beach	.0397643	.025144	.0112634	.1308403
Pier	.1597699	.0578606	.0755287	.3067894
Private	.0234452	.0042244	.0164475	.0333192
Charter	.7770206	.0620006	.6334436	.8754201
2				
mode				
Beach	.1497711	.0193181	.1157062	.1916909
Pier	.1450983	.0309011	.0943342	.2166453
Private	.5176325	.034598	.4499131	.5847104
Charter	.1874981	.0553217	.1017353	.3198181

Latent class marginal posterior probabilities					Number of obs = 1,182
	Delta-method				
	Margin	std. err.	[95% conf. interval]		
Class					
1	.3325691	.0410668	.257461	.4172751	
2	.6674309	.0410668	.5827249	.742539	

The latent class probabilities are estimated to be 0.33 and 0.67, respectively.

The `margins` command yields the AME of income on each mode of fishing.

Average marginal effects					Number of obs = 1,182
Model VCE: Robust					
dy/dx wrt: income					
1._predict: Predicted mean (1.mode), using class probabilities, predict(mu outcome(mode 1))					
2._predict: Predicted mean (2.mode), using class probabilities, predict(mu outcome(mode 2))					
3._predict: Predicted mean (3.mode), using class probabilities, predict(mu outcome(mode 3))					
4._predict: Predicted mean (4.mode), using class probabilities, predict(mu outcome(mode 4))					

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
income						
_predict						
1	-.005324	.0056746	-0.94	0.348	-.016446	.0057981
2	-.0304964	.0062628	-4.87	0.000	-.0427712	-.0182216
3	.0325652	.0071824	4.53	0.000	.018488	.0466424
4	.0032552	.0082157	0.40	0.692	-.0128474	.0193577

Only fishing modes 2 and 3 provide statistically significant (at 5%) estimates of the AME of income. An increase in income reduces the probability that mode 2 would be chosen and increases the probability that mode 3 would be chosen. Qualitatively, this result is similar to what was obtained for the (one-component) MNL model; see section [18.4.2](#). The lack of improved precision in this example is not surprising. From output not included, the two-

component model had lower AIC than the one-component model (2,958.1 versus 2,966.3) but higher BIC (3,024.1 versus 2,996.8).

23.3.4 Example 4: Tobit regression

Tobit regression is an established tool for handling censored data. The `tobit` command yields estimates of β and σ^2 in the model $y^* = \mathbf{x}'\beta + \varepsilon$, where $\varepsilon \sim N(0, \sigma^2)$ and $y = y^*$ if $y^* > 0$ and $y = 0$ otherwise.

However, tobit regression is based on the strong assumptions of normality and homoskedasticity—assumptions that frequently fail diagnostic tests. We can relax the normality assumption by replacing the standard tobit model with a mixture of tobits. This also goes some way toward relaxing the homoskedasticity assumption.

We illustrate a mixture of tobits using the same data as in section [19.3](#). The dependent variable is the level of health expenditures, with 16% of observations censored at 0. Estimation of a two-component model yields

```

. * Tobit: Two-component finite mixture tobit
. qui use mus219mepsambexp, clear
. global xlist age female educ blhisb totchr ins
. fmm 2, vce(robust) nolog: tobit ambexp $xlist, ll(0)
Finite mixture model
Log pseudolikelihood = -24776.681

```

Number of obs = 3,328

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1.Class	(base outcome)					
2.Class _cons	-1.103018	.1924958	-5.73	0.000	-1.480303	-.7257335

Class: 1

Response: ambexp

Model: tobit

Censoring of obs:

Uncensored = 2,802

Left-censored = 526

Right-censored = 0

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
ambexp						
age	98.02374	22.58601	4.34	0.000	53.75596	142.2915
female	353.825	46.33263	7.64	0.000	263.0147	444.6353
educ	33.5267	6.584909	5.09	0.000	20.62052	46.43289
blhisb	-182.4351	40.72239	-4.48	0.000	-262.2495	-102.6207
totchr	443.3924	45.40638	9.76	0.000	354.3976	532.3873
ins	34.8888	34.74336	1.00	0.315	-33.20693	102.9845
_cons	-577.927	127.9969	-4.52	0.000	-828.7962	-327.0577
var(e.ambexp)	430559.5	81254.07			297438.2	623260.7

```

Class:      2
Response: ambexp
Model:      tobit
Censoring of obs:
            Uncensored = 2,802
            Left-censored =    526
            Right-censored =      0

```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
ambexp						
age	531.4028	139.278	3.82	0.000	258.4229	804.3826
female	1046.152	388.9827	2.69	0.007	283.76	1808.544
educ	123.732	59.06497	2.09	0.036	7.966774	239.4972
blhisp	-1087.516	439.5393	-2.47	0.013	-1948.997	-226.0345
totchr	1947.121	237.1148	8.21	0.000	1482.385	2411.858
ins	-779.8928	364.4406	-2.14	0.032	-1494.183	-65.60237
_cons	-2223.764	1064.804	-2.09	0.037	-4310.741	-136.7874
var(e.ambexp)	1.89e+07	5793808			1.03e+07	3.44e+07

```
. estimates store FMM2
```

The parameter estimates differ greatly across the two components. From output not included, the first component has probability 0.751 and average predicted mean of y^* of 619, while the second component has probability 0.249 and much higher average predicted mean of 2,443.

The `margins` command provides margins for the latent variable y^* and not the observed outcome y . Because $E(y^*|\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta}$, the resulting MES from the two-component tobit model are simply the class probability weighted averages of the slope coefficients already given.

We have

```

. * Tobit: AME for two-component finite mixture tobit for E[y*|x]
. margins, dydx(*)

Average marginal effects                                         Number of obs = 3,328
Model VCE: Robust

Expression: Predicted mean (Annual ambulatory expenditures), using class
probabilities, predict(mu outcome(ambexp))
dy/dx wrt: age female educ blhisp totchr ins

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
age	206.0109	35.53168	5.80	0.000	136.37	275.6517
female	526.3354	97.20772	5.41	0.000	335.8118	716.8591
educ	56.00358	14.35708	3.90	0.000	27.86422	84.14295
blhisp	-407.9584	114.6833	-3.56	0.000	-632.7336	-183.1832
totchr	818.0837	63.75876	12.83	0.000	693.1188	943.0486
ins	-168.1342	82.63214	-2.03	0.042	-330.0902	-6.178186

Finally, we compare the goodness of fit of the two-component tobit model with the regular tobit model.

```

. * Tobit: Compare FMM2 to regular tobit using AIC and BIC
. qui tobit ambexp $xlist, ll(0) vce(robust)
. estimates store FMM1
. estimates stats FMM1 FMM2

Akaike's information criterion and Bayesian information criterion

```

Model	N	ll(null)	ll(model)	df	AIC	BIC
FMM1	3,328	-26706.46	-26359.42	8	52734.85	52783.73
FMM2	3,328	.	-24776.68	17	49587.36	49691.23

Note: BIC uses N = number of observations. See [R] BIC note.

The FMM2 tobit has greatly superior fit using the information criteria.

23.3.5 Example 5: Poisson regression

In this section, we reconsider the doctor-visits example, which appeared in section [20.5.4](#). There the application of the FMM to count regression was explored in detail. The discussion of the current application is confined to the more important features only. We compare the basic Poisson regression with FMM2 Poisson and FMM3 Poisson in terms of goodness of fit and their

implications for heterogeneity in response to a representative regressor, which we take to be `totchr`. We also calculate the latent class means using the `estat lcmean` command and the mixture probability weights using the `estat lcprob` command.

```
. * Poisson: Two-component finite mixture Poisson using count chapter data
. use mus220mepsdocvis, clear
(A.C.Cameron & P.K.Trivedi (2022): Microeconometrics Using Stata, 2e)
. global xlist private medicaid age age2 educyr actlim totchr
. qui fmm 2, vce(robust): poisson docvis $xlist
. estimates store fmm2pois
. estat lcmean
```

Latent class marginal means Number of obs = 3,677

		Delta-method				
		Margin	std. err.	z	P> z	[95% conf. interval]
1	docvis	5.050474	.2713086	18.62	0.000	4.518719 5.582229
2	docvis	11.65096	.6255685	18.62	0.000	10.42487 12.87706

```
. estat lcprob
```

Latent class marginal probabilities Number of obs = 3,677

		Delta-method		
		Margin	std. err.	[95% conf. interval]
Class	1	.6452176	.0268118	.5911008 .6958574
	2	.3547824	.0268118	.3041426 .4088992

Comparing the basic Poisson and FMM2 Poisson from output given at the end of this subsection, we see there is a major improvement in the log likelihood. The two classes have average doctor visits of 5.05 and 11.65 for the first and second latent classes, respectively. The first component, the lower usage group, has an estimated probability weight of 0.65. The second component is a high-user group, and it has probability weight of 0.35.

We next compute and report the AME of `totchr` on doctor visits from two models, FMM2 and one-component Poisson.

```

. * Poisson: ME of regressor changes in two-component model
. margins, dydx(totchr)
Average marginal effects                                         Number of obs = 3,677
Model VCE: Robust
Expression: Predicted mean (# doctor visits), using class probabilities,
            predict(mu outcome(docvis))
dy/dx wrt:  totchr

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
totchr	1.855912	.1470048	12.62	0.000	1.567788	2.144036

```

. qui poisson docvis $xlist, vce(robust)
. estimates store poisson
. margins, dydx(totchr)
Average marginal effects                                         Number of obs = 3,677
Model VCE: Robust
Expression: Predicted number of events, predict()
dy/dx wrt:  totchr

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
totchr	1.694685	.0908883	18.65	0.000	1.516547	1.872823

The AME of `totchr` is estimated to be 1.856 visits, not too different from 1.695 for the simpler Poisson model.

The detailed FMM2 Poisson coefficient estimates, not displayed above, show that the coefficient of `totchr` in the FMM2 regression is 0.329 for the first component and 0.173 for the second, so the multiplicative effect of an additional chronic condition differs across the two components. Thus, from output not listed, the ME of `totchr` evaluated at the sample means of the regressors is 4.84 for the FMM2 model and is 1.16 for the Poisson model. Hence, the mixture model is informative about heterogeneity in response between the two latent classes.

A finer distinction between the latent groups in the population adds an additional regression component. FMM3 Poisson regression yields

```
. * Poisson: Three-component finite mixture Poisson regression
. qui fmm 3, nolog vce(robust): poisson docvis $xlist
. estat lcmean
```

Latent class marginal means

Number of obs = 3,677

		Delta-method				
		Margin	std. err.	z	P> z	[95% conf. interval]
1	docvis	3.443653	.1134148	30.36	0.000	3.221365 3.665942
2	docvis	31.63433	2.238283	14.13	0.000	27.24737 36.02128
3	docvis	11.63865	.3487926	33.37	0.000	10.95503 12.32227

```
. estat lcprob
```

Latent class marginal probabilities

Number of obs = 3,677

		Delta-method		
		Margin	std. err.	[95% conf. interval]
Class				
1		.6234456	.0177598	.5880541 .6575672
2		.0391851	.0061092	.0288178 .0530781
3		.3373693	.0168279	.305217 .3710995

```
. estimates store fmm3pois
```

The first and largest class with probability 0.62 corresponds to a low-use group with an average of 3.44 doctor visits. The second and smallest class with probability 0.04 has an average of 31.63 doctor visits, and the third class with probability 0.34 has an average of 11.64 visits.

A table of information criteria is generated using the `estimates stats` command.

```
. * Poisson: Goodness-of-fit for two- and three-component models
. estimates stats poisson fmm2pois fmm3pois
Akaike's information criterion and Bayesian information criterion
```

Model	N	ll(null)	ll(model)	df	AIC	BIC
poisson	3,677	-17258.63	-15019.64	8	30055.28	30104.96
fmm2pois	3,677	.	-12100.19	17	24234.37	24339.94
fmm3pois	3,677	.	-10907.21	26	21866.42	22027.88

Note: BIC uses N = number of observations. See [R] BIC note.

The third added component improves the log likelihood enough to be preferred using the AIC and BIC criteria. Compared with the three-component model, the two-component `fmm2pois` model essentially merges the last two latent classes.

23.3.6 Example 6: Mixture distribution with point mass

In all the previously considered mixture distributions, the components were members of a common family of distributions. There is no restriction that this should always be the case. Valid mixture distributions can be constructed using members of different families of distributions.

Zero-inflated Poisson and negative binomial distributions, previously covered in section [20.6](#), are leading examples of point-mass mixtures. These models are a mixture of a binary regression (often logit) for modeling the point mass at zero and a Poisson or negative binomial regression for counts.

The following example for number of emergency room visits that are zero for most of the sample provides an `fmm` prefix equivalent to the `zinb` command for count data. The output is omitted because it is provided in section [20.6.5](#).

```
. * fmm point-mass two-component command same as zero-inflated negative binomial
. use mus220mepsemergroom, clear
. fmm: (pointmass er, lcprob(age 1.actlim totchr)) (nbreg er age 1.actlim totchr)
  (output omitted)
. zinb er age 1.actlim totchr, inflate(age 1.actlim totchr)
  (output omitted)
```

23.3.7 Example 7: Mixture application using survival data

We reconsider the model of the duration of the length of jobless spells, using data analyzed in section [21.2](#). Unobserved heterogeneity (“frailty”), usually treated as a continuous random variable, plays an important role in duration models. An FMM can be constructed using components with such a structure; a mixture of Weibull is an example.

The dependent variable `spell` is the number of periods jobless (measured in two-week intervals) and is censored from above. The regression includes `ui` (unemployment insurance status) and `logwage` as regressors. Data definitions and summary follow.

Variable name	Storage type	Display format	Value label	Variable label
<code>spell</code>	int	%8.0g		Periods jobless: two-week intervals
<code>ui</code>	int	%8.0g		Filed UI claim
<code>logwage</code>	float	%8.0g		log weekly earnings
. summarize spell ui logwage				
Variable	Obs	Mean	Std. dev.	Min
<code>spell</code>	3,343	6.247981	5.611271	1
<code>ui</code>	3,343	.5527969	.4972791	0
<code>logwage</code>	3,343	5.692994	.5356591	2.70805
				7.600402

Next, we fit the standard Weibull regression model. To save space, we do not report full output, but the AMES are shown.

```

. * Weibull: Standard Weibull regression (one component) as benchmark
. stset spell, fail(censor1=1)
Survival-time data settings
    Failure event: censor1==1
Observed time interval: (0, spell]
    Exit on or before: failure



---


    3,343  total observations
        0  exclusions


---


    3,343  observations remaining, representing
    1,073  failures in single-record/single-failure data
    20,887  total analysis time at risk and under observation
                    At risk from t =          0
                    Earliest observed entry t =      0
                    Last observed exit t =       28

. qui streg i.ui logwage, nohr vce(robust) dist(weibull) nolog
. margins, dydx(ui)

Average marginal effects                                     Number of obs = 3,343
Model VCE: Robust

Expression: Predicted median _t, predict()
dy/dx wrt: 1.ui



---



|      | Delta-method |           |       |       |                      |          |
|------|--------------|-----------|-------|-------|----------------------|----------|
|      | dy/dx        | std. err. | z     | P> z  | [95% conf. interval] |          |
| 1.ui | 14.02813     | 1.066691  | 13.15 | 0.000 | 11.93746             | 16.11881 |


```

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

The AME of having unemployment insurance (*ui*) on unemployment duration is positive.

Although the Weibull duration model is very widely used, in principle, a two-component mixture of Weibull should provide a better fit to the data. We use the *fmm* prefix to fit such a model.

```

. * Weibull: Two-component finite mixture Weibull regression
. fmm 2, nolog vce(robust): streg i.ui logwage, distribution(weibull)
Finite mixture model                                         Number of obs = 3,343
Log pseudolikelihood = -3938.0613

```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
1.Class	(base outcome)					
2.Class						
_cons	1.368373	.1815214	7.54	0.000	1.012598	1.724149

```

Class:      1
Response:   _t
Model:      streg, dist(weibull)                                     No. of failures =      1,073
                                                               Time at risk     = 20,887.00

```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
-t						
1.ui	-3.662912	.5784759	-6.33	0.000	-4.796704	-2.52912
logwage	.4706624	.2990599	1.57	0.116	-.1154842	1.056809
_cons	-3.711542	1.833207	-2.02	0.043	-7.304561	-.1185235
/_t						
ln_p	.9782369	.1106777			.7613126	1.195161

```

Class:      2
Response:   _t
Model:      streg, dist(weibull)                                     No. of failures =      1,073
                                                               Time at risk     = 20,887.00

```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
-t						
1.ui	-1.108708	.1095215	-10.12	0.000	-1.323366	-.8940497
logwage	.485504	.0881343	5.51	0.000	.3127639	.6582441
_cons	-6.509076	.583254	-11.16	0.000	-7.652233	-5.365919
/_t						
ln_p	.3034215	.0463767			.2125248	.3943182

The results are easier to interpret if we obtain latent class probabilities and means.

```
. * Weibull: Latent class probabilities and means for two-component model
. estat lcprob
```

Latent class marginal probabilities Number of obs = 3,343

	Delta-method			
	Margin	std. err.	[95% conf. interval]	
Class	1	.2028828	.0293559	.1513376 .2664717
	2	.7971172	.0293559	.7335283 .8486624

```
. estat lcmean
```

Latent class marginal means Number of obs = 3,343

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
1	_t	3.441098	.3002856	11.46	0.000	2.852549 4.029647
2	_t	24.8878	1.58575	15.69	0.000	21.77979 27.99581

The first component has probability 0.20 and a short average duration of 3.4 periods, while the second component has probability 0.80 with a much longer average of duration of 24.9 periods.

Finally, we obtain the AMES.

```
. * Weibull: MEs for two-component model
. margins, dydx(ui)
```

Average marginal effects Number of obs = 3,343
Model VCE: Robust

Expression: Predicted mean (Analysis time when record ends), using class probabilities, predict(mu outcome(_t))
dy/dx wrt: 1.ui

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.ui	15.76266	1.939978	8.13	0.000	11.96037	19.56495

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

The AME of having unemployment insurance is 15.76 periods of unemployment, compared with 14.03 periods using the standard Weibull

model. From output not given, the log likelihood rises from $-4,008.8$ to $-3,938.1$, leading to considerably lower BIC.

23.3.8 Heterogeneity and quantile regression

FMMS provide a parametric approach for modeling heterogeneous responses. In section 15.3, we used the conditional quantile regression (CQR) method, a semiparametric estimator, to analyze medical expenditures at different quantiles of expenditure distribution. We used postestimation tests of equality of regression coefficients at different quantiles. CQR is a well-established method for studying heterogeneous responses of continuous outcomes. The method has been extended to count regressions; see section [20.9](#).

23.4 Nonlinear mixed-effects models

Mixed-effects models or multilevel models or hierarchical models permit more flexible models with several levels of nesting, such as students nested in classes nested in schools. Special cases of these models include models with intercepts and slopes that may be random. This can lead to more efficient estimation, and additionally, this approach is especially advantageous in settings where heterogeneity is of intrinsic interest.

Linear mixed models can be fit using the `mixed` command, presented in section 6.7. Here we present various `me` commands that extend mixed effects to nonlinear regression models.

The `meglm` command extends mixed-effects analysis to the class of GLMs presented in section 13.3.7. Some GLMs are used so often that Stata additionally provides distinct estimation commands, rather than requiring use of the `meglm` command and specification of the relevant link function and family. These distinct commands are `melogit`, `meprobit`, `mecloglog`, `meologit`, `meprobit`, `mepoisson`, `meoprobit`, and `menbreg`.

Additional nonlinear mixed commands are `mestreg` for parametric survival models, `menl` for models with a nonlinear conditional mean and additive errors, `meintreg` for interval-regression models, and `metobit` for tobit models.

Nonlinear mixed-effects models face two challenges that are not present in the linear case. First, with rare exception, any model misspecification leads to inconsistency of parameter estimates. By contrast, in the linear case, consistency requires only correct specification of the conditional mean, though in the linear case, standard errors still need to be adjusted if the distribution for the mixed effects is misspecified. Thus, it may be best in the nonlinear case to use rich models for the mixed effects rather than, for example, having only intercepts that are random.

Second, the models are nonlinear models that can be computationally challenging to fit.

23.4.1 Mixed-effects GLM and the `meglm` command

From section 13.3.7, 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, $E(y_i|\mathbf{x}_i) = \exp(\mathbf{x}'_i \boldsymbol{\beta})$, and the link function $g(\cdot)$ is $\ln(\cdot)$, the inverse of the function.

We consider a two-level mixed-effects GLM for simplicity with individual i in group j . Then, we add the mixed-effects term $\mathbf{z}'_{ij} \mathbf{u}_j$, where \mathbf{z}_{ij} are observable variables and \mathbf{u}_j is a vector of i.i.d. normally distributed random effects. We have

$$E(y_{ij}|\mathbf{x}_{ij}, \mathbf{z}_{ij}, \mathbf{u}_j) = g^{-1}(\mathbf{x}'_{ij} \boldsymbol{\beta} + \mathbf{z}'_{ij} \mathbf{u}_j)$$

where $\mathbf{u}_j \sim N(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{u}})$ and the specified link function $g(\cdot)$ differs for different GLMS.

Specific choices of \mathbf{z}_{ij} lead to some standard models. A regular GLM corresponds to $\mathbf{z}_{ij} = \mathbf{0}$. A random-effects model with random intercept corresponds to $\mathbf{z}_{ij} = 1$ because only the intercept is random, and then $\mathbf{z}'_{ij} \mathbf{u}_j = u_j$. A model often called the random-coefficients model sets $\mathbf{z}_{ij} = \mathbf{x}_{ij}$ so that, for the regressor \mathbf{x}_{ij} , both the intercept and slope coefficients are random.

Denote the conditional density for an individual observation as $f(y_{ij}|\eta_{ij})$, where $\eta_{ij} = \mathbf{x}'_{ij} \boldsymbol{\beta} + \mathbf{z}'_{ij} \mathbf{u}_j$. For example, for the Poisson model, $f(y_{ij}|\eta_{ij}) = \exp(-\eta_{ij})\eta_{ij}^{y_{ij}}/y_{ij}!$ The log density for all N_j individuals in group j is $\sum_{j=1}^{N_j} \ln f(y_{ij}|\eta_{ij})$. The problem is that this log density includes the random effects \mathbf{u}_j that need to be integrated out. The log likelihood for cluster j upon integrating out these unobservables is then

$$L_j(\boldsymbol{\beta}, \boldsymbol{\Sigma}_{\mathbf{u}}) = \int \left[\sum_{j=1}^{N_j} \ln f\{y_{ij}|g^{-1}(\mathbf{x}'_{ij} \boldsymbol{\beta} + \mathbf{z}'_{ij} \mathbf{u}_j)\} \right] \times \phi(\mathbf{u}_j|\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{u}}) d\mathbf{u}_j \quad (23.2)$$

where $\phi(\mathbf{z}|\mathbf{0}, \boldsymbol{\Sigma})$ denotes the density of the multivariate normal with mean $\mathbf{0}$ and variance $\boldsymbol{\Sigma}$.

Consistent estimation of β will require correct specification of all three of the density $f(\cdot)$, the link function $g(\cdot)$, and the distribution of u_j . By contrast, in the simpler GLM without mixed effects, or in the linear mixed-effects model, consistent estimation of β requires only correct specification of the conditional mean, that is, of the link function. Unlike the linear case, the integral in (23.2) is multidimensional with no closed-form solution. Numerical computation can be challenging, especially if u_{ij} is high dimensional.

The `meglm` command has format similar to the `mixed` command presented in section 6.7.3. The specific GLM model fit is defined using the `family()` and `link()` options.

Also, `meglm` provides six numerical integration methods:

1. the mean-and-variance adaptive Gauss–Hermite quadrature (`gsem` option `intmethod(mvaghermite)`, which is the default);
2. the mode-and-curvature adaptive Gauss–Hermite quadrature (`intmethod(mcaghermite)`);
3. the Pinheiro–Chao mode-curvature adaptive Gauss–Hermite quadrature (`intmethod(pcaghermite)`);
4. the nonadaptive Gauss–Hermite quadrature (`intmethod(ghermite)`);
5. the Laplacian approximation (`intmethod(laplace)`); and
6. the Pinheiro–Chao Laplacian approximation (`intmethod(pclaplace)`).

See section 5.5.1 for a brief introduction.

Methods 1 and 2 are preferred, while methods 3 to 6 may provide better starting values for methods 1 and 2 if there are convergence problems. Techniques `pcaghermite` and `pclaplace` can speed up computation but are available only with `family(binomial)` and `family(bernoulli)` combined with `link(logit)` and with `family(poisson)`. Option `intpoints(#)` enables changing the number of evaluation points in methods 1–4 from the default of 7. Choosing a lower number speeds up model fitting and improves convergence somewhat but at the expense of less accuracy.

23.4.2 Poisson and negative binomial mixed-effects example

As a nonlinear two-level mixed-effects example, we consider Poisson regression using a dataset with clustering introduced in section 6.4. The

dependent variable is the number of pharmacy visits (`pharvis`), and the regressors are an indicator for illness (`illness`) and log total household expenditure (`lnhhexp`).

```
. * Read in data and drop a few observations
. qui use mus206vlss, clear
. drop if missing(lnhhexp) | (lnhhexp > 2.579681 & lnhhexp < 2.579683)
(13 observations 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

Individuals live in one of 194 communes, and we expect individual responses to vary by commune. So we specify `commune` as a second level and allow the intercept and the coefficient of `illness` to vary at this second level. Then y_{ij} has conditional mean

$$\exp\{(\beta_1 + u_{1j}) + (\beta_2 + u_{2j}) \times \text{illness}_{ij} + \beta_3 \times \text{lnhhexp}_{ij}\}$$

where u_{1j} and u_{2j} are zero-mean normally distributed random variables that are independently drawn across categories j that correspond to different ages (measured in years).

The mixed-effects Poisson model can be fit using the `meglm` command with options `family(poisson)` and `link(log)`. Alternatively, and more simply, the equivalent `mepoisson` command can be used. We obtain

```

. * Mixed model: Poisson, 2nd level commune, coeff of intercept and illness vary
. mepoisson pharvis lnhhexp illness || commune: illness, vce(robust)

Fitting fixed-effects model:
Iteration 0: log likelihood = -71403.382
Iteration 1: log likelihood = -26812.91
Iteration 2: log likelihood = -26219.27
Iteration 3: log likelihood = -26217.797
Iteration 4: log likelihood = -26217.797

Refining starting values:
Grid node 0: log likelihood = -23897.571

Fitting full model:
Iteration 0: log pseudolikelihood = -23897.571 (not concave)
Iteration 1: log pseudolikelihood = -23867.866 (not concave)
Iteration 2: log pseudolikelihood = -23840.085 (not concave)
Iteration 3: log pseudolikelihood = -23818.108
Iteration 4: log pseudolikelihood = -23803.899
Iteration 5: log pseudolikelihood = -23777.049
Iteration 6: log pseudolikelihood = -23759.62
Iteration 7: log pseudolikelihood = -23759.194
Iteration 8: log pseudolikelihood = -23759.192

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)      =    261.87
Log pseudolikelihood = -23759.192
Prob > chi2       = 0.0000
(Std. err. adjusted for 194 clusters in commune)

```

pharvis	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
lnhhexp	-.1287277	.0410759	-3.13	0.002	-.209235 -.0482204
illness	.9563129	.0613635	15.58	0.000	.8360427 1.076583
_cons	-1.305269	.1289779	-10.12	0.000	-1.558061 -1.052477
commune					
var(illness)	.1797316	.0520961			.1018356 .3172119
var(_cons)	.404467	.0681602			.2906965 .562764

The first iterations that are described as fitting what the mixed-models literature calls a fixed-effects model simply perform the standard Poisson regression. Subsequent iterations are initially over a nonconcave portion of the objective function. Here the defaults of mean-and-variance adaptive Gauss–Hermite quadrature with seven evaluation points are used. The default for the `mepoisson` command is to restrict the two random effects to be independent of each other;

the option `covariance(unstructured)` allows the random effects to be correlated.

There is substantial variation across communes in the coefficient of `illness` because the commune-specific random effect of `illness` has standard deviation of $\sqrt{0.1797316} = 0.424$ compared with the fixed coefficient of `illness` that has value of 0.956.

A Wald test of the joint statistical significance of the random coefficients, a joint test of whether $\text{Var}(u_{1j}) = 0$ and $\text{Var}(u_{2j}) = 0$, yields

```
. * Wald test of the random coefficients
. test ([/]var(illness[commune])=0) ([/]var(_cons[commune])=0)
( 1) [/]var(illness[commune]) = 0
( 2) [/]var(_cons[commune]) = 0
      chi2( 2) =    73.25
      Prob > chi2 =    0.0000
```

Because of the one-sided nature of the test (the variances must be greater than zero), the true distribution of this test statistic is complicated. Using $\chi^2(2)$ critical values provides an upper bound to the reported *p*-value, so the test is conservative. Because *p* = 0.000, we definitely find statistical significance at level 0.05.

We next fit a Poisson model and ME Poisson and ME negative binomial models with different estimates of standard errors.

```
. * Mixed model: Compute various models and standard errors
. qui poisson pharvis lnhhexp illness, vce(cluster commune)
. estimates store p_clu
. qui mepoisson pharvis lnhhexp illness || commune: illness
. estimates store mep_def
. qui mepoisson pharvis lnhhexp illness || commune: illness, vce(cluster commune)
. estimates store mep_rob
. qui menbreg pharvis lnhhexp illness || commune: illness
. estimates store menb_def
. qui menbreg pharvis lnhhexp illness || commune: illness, vce(robust)
. estimates store menb_rob
```

The following table presents the results:

```
. * Mixed model: Compare with standard Poisson and get robust standard errors
. estimates table p_clu mep_def mep_rob menb_def menb_rob,
> eq(1) b(%8.4f) se stfmt(%8.1f) stats(l1 N aic chi2_c df_c)
```

Variable	p_clu	mep_def	mep_rob	menb_def	menb_rob
#1					
lnhhexp	0.0175 0.0473	-0.1287 0.0188	-0.1287 0.0411	-0.0875 0.0277	-0.0875 0.0405
illness	0.6806 0.0287	0.9563 0.0319	0.9563 0.0614	1.3292 0.0461	1.3292 0.0575
_cons	-1.4121 0.1451	-1.3053 0.0681	-1.3053 0.1290	-1.9047 0.0878	-1.9047 0.1248
var(il~ss[com~e])		0.1797 0.0254	0.1797 0.0521	0.3216 0.0438	0.3216 0.0538
var(_cons[com~e])		0.4045 0.0497	0.4045 0.0682	0.3542 0.0480	0.3542 0.0476
/lnalpha				0.1792 0.0276	0.1792 0.0603
Statistics					
l1	-26217.8	-23759.2	-23759.2	-21116.2	-21116.2
N	27753	27753	27753	27753	27753
aic	52441.6	47528.4	47528.4	42244.5	42244.5
chi2_c		4917.2	4917.2	1753.1	1753.1
df_c		2.0	2.0	2.0	2.0

Legend: b/se

Comparing the first and second columns, we see considerable difference in the estimates of the main model slope parameters. Unlike linear models, introducing random effects in a nonlinear model changes the functional form for $E(y_{ij}|\mathbf{x}_{ij})$.

The `vce(robust)` option of the `mepoisson` command obtains cluster-robust standard errors that cluster on `commune`. Comparing the second and third columns, we see that the default standard errors are greatly understated and the `vce(robust)` option should be used. The fourth and fifth columns provide estimates for a mixed negative binomial model, using the `nbreg` command, and even though the negative binomial may better control for the overdispersion typical of count data, the `vce(robust)` option should be used. The information criteria strongly favor the mixed negative binomial model.

23.4.3 Prediction and MEs

The random effects are often of intrinsic interest, and even if that is not the case, computation of conditional mean predictions and MES requires control for the random effects.

We first illustrate `predict`. The `reffects` and `ebmeans` options obtain predictions of the two random effects for each of the 194 communes using the posterior means of the random effects. The `mu` and `conditional()` options obtain the conditional mean $E(y_{ij}|\mathbf{x}_{ij}, \mathbf{z}_{ij}, \mathbf{u}_j^*)$, where \mathbf{u}_j^* may be set to 0 (`conditional(fixedonly)`), set to the posterior means (`conditional(ebmeans)`), or set to the posterior modes (`conditional(ebmodes)`). The `mu` and `marginal` options numerically integrates out the random effects and computes $E(y_{ij}|\mathbf{x}_{ij}, \mathbf{z}_{ij})$. The formulas for these methods are detailed in *Methods and formulas* in [ME] **meglm postestimation**. We have

```
. * Various predictions of random effects and of mean
. estimates restore mep_rob
(results mep_rob are active now)

. predict re_ebmeans*, reffects ebmeans
(calculating posterior means of random effects)
(using 7 quadrature points)

. predict mu_fixed, mu conditional(fixedonly)
. predict mu_ebmeans, mu conditional(ebmeans)
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)

. predict mu_ebmode, mu conditional(ebmode)
(predictions based on fixed effects and posterior modes of random effects)
. predict mu_marg, mu marginal
. summarize re_ebmeans* pharvis mu_fixed mu_ebmeans mu_ebmode mu_marg
```

Variable	Obs	Mean	Std. dev.	Min	Max
re_ebmeans1	27,753	-.0551694	.3669602	-.6321057	1.159421
re_ebmeans2	27,753	-.0723159	.6324277	-1.792576	1.307337
pharvis	27,753	.5110439	1.312606	0	30
mu_fixed	27,753	.694688	6.587993	.1351856	1023.213
mu_ebmeans	27,753	.506191	.8097099	.0330438	32.45189
mu_ebmode	27,753	.5121716	.8178656	.0349858	32.58508
mu_marg	27,753	69.43721	10901.27	.1654853	1815856

The first two rows list the posterior mean estimates of the random effects for the intercept and for `illness`. Prediction of the mean evaluated at either posterior means or posterior modes of the random effects gives an average prediction

close to the sample average of the dependent variable, while prediction setting the random effects to 0 in this nonlinear model does not, because $E(y_{ij}|\mathbf{x}_{ij}, \mathbf{z}_{ij}, \mathbf{u}_j = \mathbf{0}) = \exp(\mathbf{x}'_{ij}\boldsymbol{\beta}) \neq E(y_{ij}|\mathbf{x}_{ij}, \mathbf{z}_{ij})$. The final row shows that there is clearly a problem in this example with estimates of $E(y_{ij}|\mathbf{x}_{ij}, \mathbf{z}_{ij})$ that differ greatly from the estimate of the sample average of the dependent variable. This is not pursued further for this pedagogical example.

For margins and MES, interest lies in $E(y_{ij}|\mathbf{x}_{ij}, \mathbf{z}_{ij})$, which can be obtained using the `marginal` option of the `margins` postestimation command. The alternative is the `conditional(fixedonly)` option that uses $E(y_{ij}|\mathbf{x}_{ij}, \mathbf{z}_{ij}, \mathbf{u}_j = \mathbf{0})$. As already noted, prediction using the `marginal` option had problems, so instead, for illustrative purposes, we compute AMES when the random effects are set to 0. We have

```
. margins, dydx(*) predict(mu conditional(fixedonly))
Average marginal effects                                         Number of obs = 27,753
Model VCE: Robust
Expression: Predicted mean, fixed portion only, predict(mu
            conditional(fixedonly))
dy/dx wrt: lnhhexp illness
```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
lnhhexp	-.0894256	.0328357	-2.72	0.006	-.1537824	-.0250687
illness	.6643391	.1717332	3.87	0.000	.3277483	1.00093

23.5 Linear structural equation models

Structural equation models (SEMS) describe a very wide range of models, including standard regression models and standard latent variable models. The SEM framework goes by various names in various disciplines, including covariance structure analysis, causal models, path analysis, and latent variable modeling. Unlike FMMs and mixed models, SEMs handle endogenous regressors.

Models that fall into the SEM class and that can be estimated using the `sem` command, with the usual Stata command given in parentheses, include linear regression (`regress`) and seemingly unrelated equations (`sureg`); latent variable models such as linear mixed models (`mixed`) and factor analysis (`factor`); linear models with endogenous regressors such as single-equation instrumental-variables (IV) (`ivregress`) and multiequation simultaneous equations models (`reg3`).

Stata provides two commands. Command `sem` is for linear models and enables estimation of richer variants of the preceding models, with endogenous variables and latent variables, and permits estimation of some basic models for which there are no specific Stata commands, such as measurement-error models.

Command `gsem` for generalized structural equation models (GSEMs) extends the methods to a wide class of nonlinear models, allows use of factor-variable notation, and can be used to estimate structural systems of nonlinear equations. For example, it provides a structural way to specify and fit a count model with an endogenous regressor. The `gsem` command is presented in section [23.6](#).

In both cases, SEMs place restrictions on the mean vector and covariance matrix of observable variables. Estimation is then by ML under the assumption of joint normality of observables and unobservable latent variables. For linear SEMs, an alternative estimator relaxes the normality assumption to obtain parameter estimates that set sample first and second moments close to the first and second moments implied by the SEM.

Several caveats are in order. First, it is possible to specify models that are not identified, such as in the well-known case of simultaneous equations models.

Second, SEMs involve strong parametric assumptions on unobservables. Linear SEMs are robust to nonnormality of observable variables, provided their mean vector and variance matrix is correctly specified. But SEMs that include latent variables may be inconsistently estimated if the latent variables are not normally distributed. And for GSEMs, nonnormality leads to estimator inconsistency.

Third, estimation of more complex models, particularly GSEMs, can be computationally challenging, and some additional effort, such as choosing good starting values, may be necessary in order for estimators to converge.

The SEM framework is used extensively in social sciences other than economics. An attraction to many practitioners is that models can be represented without the use of algebra. Instead, models are specified using path diagrams that outline the links among and between observed variables and latent variables. Stata provides an interactive SEM builder that facilitates drawing path analysis diagrams.

In this section, we provide an introduction to SEMs oriented to econometricians unfamiliar with SEMs, followed by a measurement-error example that introduces a latent variable, and a brief discussion of models with endogenous regressors. We do not cover use of SEM for linear mixed models. We conclude with a general presentation of the estimation methods.

23.5.1 SEMs for linear regression

We explain the SEM approach using the familiar linear regression model

$$y_i = \alpha + \mathbf{x}'_i \boldsymbol{\beta} + \varepsilon_i$$

where here the intercept is deliberately treated separately from the slope coefficients, \mathbf{x}_i is a $(K - 1) \times 1$ vector, and we assume independent

observations.

The usual approach taken by econometricians is to model y_i conditional on \mathbf{x}_i . Estimation is based either on the conditional mean $E(y_i|\mathbf{x}_i)$ and conditional variance $\text{Var}(y_i|\mathbf{x}_i)$ or on the conditional density $f(y_i|\mathbf{x}_i)$.

The SEM approach instead models (y_i, \mathbf{x}_i) jointly. Estimation is based either on the unconditional means [$E(y_i)$ and $E(\mathbf{x}_i)$] and variances and covariance of y_i and \mathbf{x}_i or on the unconditional joint density $f(y_i, \mathbf{x}_i)$.

Conditional on regressors approach

Taking a conditional mean and variance approach, and assuming homoskedastic errors, a moment-based approach yields the ordinary least-squares (OLS) estimator, with

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = \left\{ \sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{x}}) (\mathbf{x}_i - \bar{\mathbf{x}})' \right\}^{-1} \sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{x}}) (y_i - \bar{y})$$

$$\hat{\alpha}_{\text{OLS}} = \bar{y} - \bar{\mathbf{x}}' \hat{\boldsymbol{\beta}}$$

A fully parametric conditional approach specifies the conditional distribution of y_i given \mathbf{x}_i . If $\varepsilon_i|\mathbf{x}_i \sim N(0, \sigma^2)$, then $y_i|\mathbf{x}_i \sim N(\alpha + \mathbf{x}'_i \boldsymbol{\beta}, \sigma^2)$. The ML estimates $\hat{\alpha}$, $\hat{\boldsymbol{\beta}}$, and $\hat{\sigma}^2$ maximize the associated log-likelihood function. The ML estimates of $\hat{\alpha}$ and $\hat{\boldsymbol{\beta}}$ turn out to be identical to the OLS estimates, while $\hat{\sigma}^2 = 1/N \sum_{i=1}^N (y_i - \hat{\alpha} - \mathbf{x}'_i \hat{\boldsymbol{\beta}})^2$.

Moment-based estimation for SEMs

The moment-based variant of the SEM approach is based on mean and variance assumptions. We add the assumption that \mathbf{x}_i has mean $\mu_{\mathbf{x}}$ and variance matrix $\Sigma_{\mathbf{xx}}$. As before, we assume that $y_i|\mathbf{x}_i$ has conditional mean $\alpha + \mathbf{x}'_i \boldsymbol{\beta}$ and conditional variance σ^2 .

These assumptions imply that unconditionally, y_i has mean $\mu_y = \alpha + \mu'_{\mathbf{x}} \boldsymbol{\beta}$ and variance $\sigma_{yy} = \boldsymbol{\beta}' \Sigma_{\mathbf{xx}} \boldsymbol{\beta} + \sigma^2$ and that $\text{Cov}(y_i, \mathbf{x}_i) = \Sigma_{\mathbf{xx}} \boldsymbol{\beta}$. It follows that the covariance matrix of (y_i, \mathbf{x}_i) is

$$\Sigma \equiv \begin{bmatrix} \sigma_{yy} & \Sigma_{yx} \\ \Sigma_{xy} & \Sigma_{xx} \end{bmatrix} = \begin{bmatrix} \beta' \Sigma_{xx} \beta + \sigma^2 & \beta' \Sigma_{xx} \\ \Sigma_{xx} \beta & \Sigma_{xx} \end{bmatrix} \quad (23.3)$$

The lower-left entry in (23.3) implies that $\Sigma_{xy} = \Sigma_{xx} \beta$ or that $\beta = \Sigma_{xx}^{-1} \Sigma_{xy}$. If we replace the covariance matrices Σ_{xx} and Σ_{xy} by their usual sample estimates, we obtain

$$\hat{\beta} = \left\{ \frac{1}{N-1} \sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{x}}) (\mathbf{x}_i - \bar{\mathbf{x}})' \right\}^{-1} \frac{1}{N-1} \sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{x}}) (y_i - \bar{y})$$

But this is just the OLS estimator of β . The upper-left entry in (23.3) implies that $\sigma^2 = \sigma_{yy} - \beta' \Sigma_{xx} \beta$, which after some algebra can be shown to yield $\hat{\sigma}^2 = 1/N \sum_{i=1}^N (y_i - \hat{\alpha} - \mathbf{x}'_i \hat{\beta})^2$. The mean vector of (y_i, \mathbf{x}_i) is

$$\mu \equiv \begin{bmatrix} \mu_y \\ \mu_x \end{bmatrix} = \begin{bmatrix} \alpha + \mu'_x \beta \\ \mu_x \end{bmatrix} \quad (23.4)$$

The upper entry implies $\alpha = \mu_y - \mu'_x \beta$. If we replace the means μ_y and μ_x by their usual sample estimates, we obtain $\hat{\alpha} = \bar{y} - \bar{\mathbf{x}}' \hat{\beta}$.

To summarize, the restrictions on the covariance matrix of (y_i, \mathbf{x}_i) led to estimates of β and σ^2 , and restrictions on the mean of (y_i, \mathbf{x}_i) led to an estimate of α . The Stata documentation refers to this moment-based SEM approach as an asymptotic distribution free method because consistency requires only correct specification of means, variance, and covariances.

Maximum likelihood estimation for SEMs

A second SEM approach, the `sem` default, is fully distributional. We suppose that (y_i, \mathbf{x}_i) is joint normally distributed with the variance matrix given in (23.3).

Define the K -dimensional data vector $\mathbf{z}_i = (y_i, \mathbf{x}'_i)$ and the $(K + 1)$ -dimensional parameter vector $\boldsymbol{\theta} = (\beta, \alpha, \sigma^2)$, and let $\Sigma(\boldsymbol{\theta})$ denote the final matrix in (23.3) and $\mu(\boldsymbol{\theta})$ denote the final vector in (23.4). Then the log-likelihood function for N independent observations is

$$\begin{aligned}\ln L(\boldsymbol{\theta}) = & -\frac{1}{2} \left[NK \ln 2\pi + N \ln |\Sigma(\boldsymbol{\theta})| \right. \\ & \left. + \sum_{i=1}^N \{\mathbf{z}_i - \mu(\boldsymbol{\theta})\} \Sigma(\boldsymbol{\theta})^{-1} \{\mathbf{z}_i - \mu(\boldsymbol{\theta})\}' \right] \end{aligned}\quad (23.5)$$

Minimization with respect to $\boldsymbol{\theta} = (\beta, \alpha, \sigma^2)$ can be shown to again yield the OLS estimates $\hat{\boldsymbol{\beta}}_{OLS}$ and $\hat{\alpha}_{OLS}$, and $\hat{\sigma}^2 = (1/N) \sum_{i=1}^N (y_i - \hat{\alpha} - \mathbf{x}'_i \hat{\boldsymbol{\beta}})^2$.

While joint normality is assumed to obtain $\ln L(\boldsymbol{\theta})$ in (23.5), the estimator is a quasi-maximum likelihood estimation (MLE) that is consistent provided only that $\mu(\boldsymbol{\theta})$ and $\Sigma(\boldsymbol{\theta})$ are correctly specified. Heteroskedastic-robust and cluster-robust standard errors can be used rather than default standard errors that assume homoskedasticity.

23.5.2 SEM linear regression example

Before providing more details on the `sem` command, we provide a linear regression example, using the same doctor-visits example as in section 20.3 but treating the dependent variable `docvis` as a continuous variable, rather than as a count. In section 23.6, we will use the `gsem` command and analyze doctor visits as a count.

Using the `sem` command to fit a linear regression model, we obtain

```

. * SEM method ml: Linear regression example
. qui use mus220mepsdocvis, clear
. sem (docvis <- private medicaid age age2 educyr actlim totchr)
Endogenous variables
    Observed: docvis
Exogenous variables
    Observed: private medicaid age age2 educyr actlim totchr
Fitting target model:
Iteration 0:  log likelihood = -64884.907
Iteration 1:  log likelihood = -64884.907
Structural equation model                                         Number of obs = 3,677
Estimation method: ml
Log likelihood = -64884.907

```

	OIM					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
Structural						
docvis						
private	.9193268	.2487285	3.70	0.000	.4318279	1.406826
medicaid	.6333396	.3417109	1.85	0.064	-.0364015	1.303081
age	1.963495	.4464282	4.40	0.000	1.088512	2.838478
age2	-.0129913	.0029699	-4.37	0.000	-.0188121	-.0071705
educyr	.1915361	.0321189	5.96	0.000	.1285842	.2544879
actlim	1.369498	.2675973	5.12	0.000	.8450171	1.893979
totchr	1.883071	.0892863	21.09	0.000	1.708073	2.058069
_cons	-73.44753	16.68711	-4.40	0.000	-106.1537	-40.7414
var(e.docvis)	45.86223	1.069605			43.81304	48.00727

LR test of model vs. saturated: chi2(0) = 0.00 Prob > chi2 = .

The `sem` command distinguishes between endogenous and exogenous variables; here the dependent variable `docvis` is the endogenous variable. The default estimation method for the `sem` command is MLE under joint normality of all observable variables, here `docvis` and the seven regressors. The default standard errors use minus the inverse of the observed Hessian. The likelihood-ratio (LR) test given at the end is appropriate if the model is overidentified. Here the model is just identified, so the test is not applicable.

The following code compares 1) `sem` MLE (default method `ml`) with default variance-covariance matrix of the estimator (VCE); 2) `sem` MLE with robust VCE; 3) `sem` estimated using just mean and variance restrictions (method `adf`); 4) OLS with default VCE; and 5) OLS with robust VCE. Here by

“robust data-generating process” (DGP), we mean heteroskedastic robust. A robust VCE is not available for `sem` with method `adf`.

```
. * SEM methods compared with OLS: Linear regression example
. qui sem (docvis <- private medicaid age age2 educyr actlim totchr)
. estimates store SEMMLE
. qui sem (docvis <- private medicaid age age2 educyr actlim totchr), vce(robust)
. estimates store SEMMLErob
. qui sem (docvis <- private medicaid age age2 educyr actlim totchr), method(adf)
. estimates store SEMADF
. qui regress docvis private medicaid age age2 educyr actlim totchr
. estimates store OLS
. qui regress docvis private medicaid age age2 educyr actlim totchr, vce(robust)
. estimates store OLSrob
```

We obtain the following parameter estimates, standard errors, and associated diagnostic statistics.

```
. * SEM methods compared with OLS: Table of estimates
. estimates table SEMMLE SEMMLErob SEMADF OLS OLSrob,
> eq(1) b(%10.4f) se stfmt(%10.2f) stats(rmse r2 ll critvalue N)
```

Variable	SEMMLE	SEMMLErob	SEMADF	OLS	OLSrob
#1					
private	0.9193 0.2487	0.9193 0.2386	0.9193 0.2374	0.9193 0.2490	0.9193 0.2388
medicaid	0.6333 0.3417	0.6333 0.4090	0.6333 0.4067	0.6333 0.3421	0.6333 0.4094
age	1.9635 0.4464	1.9635 0.4049	1.9635 0.4021	1.9635 0.4469	1.9635 0.4052
age2	-0.0130 0.0030	-0.0130 0.0027	-0.0130 0.0027	-0.0130 0.0030	-0.0130 0.0027
educyr	0.1915 0.0321	0.1915 0.0318	0.1915 0.0315	0.1915 0.0322	0.1915 0.0318
actlim	1.3695 0.2676	1.3695 0.2946	1.3695 0.2934	1.3695 0.2679	1.3695 0.2948
totchr	1.8831 0.0893	1.8831 0.1017	1.8831 0.1011	1.8831 0.0894	1.8831 0.1018
_cons	-73.4475 16.6871	-73.4475 15.1800	-73.4475 15.0763	-73.4475 16.7053	-73.4475 15.1945
var(e.docvis)	45.8622 1.0696	45.8622 5.9428	45.8622 5.8999		
Statistics					
rmse				6.78	6.78
r2				0.16	0.16
ll	-64884.91	-64884.91		-12250.88	-12250.88
critvalue	-64884.91	-64884.91	0.00		
N	3677	3677	3677	3677	3677

Legend: b/se

As expected, for linear regression, the two SEM methods of estimation are equivalent to OLS, so coefficient estimates are the same in all five columns. The default standard errors for the two SEM methods and OLS differ very slightly because of different degrees-of-freedom corrections. Similarly, the two sets of robust standard errors differ very slightly because of different degrees-of-freedom corrections.

The SEM estimate of the model error variance σ^2 is 45.8622. For OLS, the rmse is an estimate of σ and equals 6.7795. Then $6.7795^2 = 45.9616$, which differs slightly from 45.8622 because of different degrees-of-freedom correction.

The log likelihoods following the `sem` and `regress` commands differ greatly. This is because the `regress` log likelihood is based on the conditional distribution of y_i given \mathbf{x}_i , whereas the ML version of `sem` log likelihood is based on the joint distribution of y_i and \mathbf{x}_i .

The `adf` method for the `sem` command minimizes the objective function defined in (23.7) below. For a just-identified model, the case here, this objective function attains a minimum value of zero.

23.5.3 SEM model specification

SEMs have three components: 1) observed variables; 2) latent variables; and 3) model errors.

A latent variable is a variable that is not observed. Usually, it is a variable that ideally we would observe. For example, in a log-earnings regression, the latent variable might be true years of schooling, while observed schooling is measured with error. A model error is a special kind of latent variable that is purely random.

A variable is exogenous if it is determined outside the system, so that no variable determines it but it may determine other variables. A variable is endogenous if it is determined by the system, so that it is determined in part by variables in the system. Observed variables and latent variables can be exogenous or endogenous, while model errors are always exogenous. Note that the SEM use of “endogenous” differs from the econometricians’ use of the term. In particular, an SEM endogenous regressor need not necessarily lead to endogeneity bias.

An SEM can be represented mathematically. Then an endogenous variable is one that at some point appears on the left-hand side of an equation, while exogenous variables always appear on the right-hand side.

An alternative way to represent an SEM, one very popular in the social sciences, is using a path diagram. Then arrows are used to represent relationships between variables. A variable is endogenous if any paths (arrows) point to it; otherwise, it is exogenous. In a path diagram, observed

variables are represented using boxes, while latent variables are represented using circles.

23.5.4 The `sem` command

The `sem` command is very complicated. We provide a very brief summary. The syntax is

```
sem paths [if] [in] [weight] [, options]
```

where the *path* defines the SEM in command-language path notation; examples are given below.

The default is estimation by ML (method `ml`), in which case the `vce()` options include `vce(robust)` and `vce(cluster clustvar)`. An alternative is moment-based estimation (method `adf`), which does not have the preceding `vce()` options and instead computes standard errors based on the observed Hessian (the default) or the expected Hessian. Options `vce(bootstrap)` and `vce(jackknife)` are also available.

The basic `sem` command has default treatments of covariances, variances, and means of latent variables. The `covariance()`, `variance()`, and `mean()` options allow these to be overridden.

Latent variables are given names that must begin with a capital letter, while the names of observed variables must begin with a lowercase letter. For example, if variable *x* is observed, it is denoted `x`, while if it is a latent variable, it is denoted `X`.

Every endogenous variable, whether latent or observed, has an error attached with it that has prefix `e`. For example, the endogenous variable *y* has error `e.y`, and the latent variable *L* has error `e.L`.

Latent variables require a normalization because these variables are not naturally scaled. These normalizations are presented in section [23.5.10](#).

If variable *x* determines *y*, we can write either `(y <- x)` or `(x -> y)`. Paths can be combined so, for example, `(y <- x1) (y <- x2)` is equivalent

```
to (y <- x1 x2).
```

The default is to assume that 1) all exogenous variables (observed or latent) are correlated with each other; 2) all error variables are uncorrelated with each other; and 3) endogenous variables are not directly correlated with each other, though they may be via their error. These defaults can be overridden using the `covariance()` option. For example, `cov(name1 name2)` is used to specify that `name1` and `name2` are uncorrelated variables, while `cov(name1*name2@0)` is used if they are correlated.

The default is to assume that 1) all observed exogenous variables have nonzero mean; 2) all latent exogenous variables have zero mean; 3) all error variables have mean zero; and 4) endogenous variables have no separate mean. The default for latent exogenous variables and for observed exogenous variables in the special case of option `noxconditional` can be overridden using the `mean()` option.

Constraints can be introduced using @ (the “at” symbol). An exact value can be given, such as `L@1`, or a symbolic such as `L@c` if, for example, an equality constraint is used.

Equations for observed endogenous variables are assumed to have a constant. For latent endogenous variables, there is not a constant unless you ask for it.

A recursive model is one that does not have any feedback loops, such as `(y1 <- y2)` and `(y2 <- y1)`, and does not have any error correlation, such as `covar(e.y1*e.y2)`. Recursive models are simple to fit and do not suffer from potential simultaneity bias. Otherwise, the model is nonrecursive.

With K observed variables, there are $K \times (K + 1)/2$ unique entries in the sample covariance. Let p be the number of model parameters, excluding the intercepts and means. If $p > K(K + 1)/2$, the model is not identified, and parameter estimates are meaningless. If $p = K \times (K + 1)/2$, the model is just identified. If $p < K \times (K + 1)/2$, the model is overidentified, and one can compute an overidentifying restrictions test.

23.5.5 Some sem command examples

Table 23.1 presents `sem` commands that yield the same results as some leading standard Stata estimation commands. Additional examples include linear mixed models.

Table 23.1. `sem` commands and equivalent standard Stata commands

Estimator	Usual command and <code>sem</code> equivalent
Sample mean	<code>mean y</code> <code>sem y</code>
Correlation	<code>correlate y x</code> <code>sem y x, standardized</code>
<i>t</i> test	<code>ttest y, by(d)</code> <code>sem y <- d</code>
OLS regression	<code>regress y x1 x2</code> <code>sem y <- x1 x2</code>
Multivariate regression	<code>mvreg y1 y2 = x1 x2</code> <code>sem y1 y2 <- x1 x2, cov(e.y1*e.y2)</code>
Seemingly unrelated regressions	<code>sureg (y1 x1 x2) (y2 x1 x3), isure</code> <code>sem (y1 <- x1 x2) (y2 <- x1 x3), cov(e.y1*e.y2)</code>
SUR with parameter constraints	<code>constraint 1 [y1]x1 = [y2]x1</code> <code>sureg (y1 x1 x2) (y2 x1 x3), constraint(1)</code> <code>sem (y1 <- x1@ a x2) (y2 <- x1@ a x3), ///</code> <code>cov(e.y1*e.y2)</code>
IV regression	<code>ivregress 2sls y1 (y2 = x2) x1</code> <code>sem (y1 <- y2 x1) (y2 <- x1 x2), ///</code> <code>cov(e.y1*e.y2)</code>
Three-stage least squares (3SLS)	<code>reg3 (y1 = y2 x1 x2 x3) (y2 = y1 x1 x4), sure</code> <code>sem (y1 <- y2 x1 x2 x3) (y2 <- y1 x1 x4), ///</code> <code>cov(e.y1*e.y2)</code>

For SUR and two-stage least squares (2SLS), the default for the `sem` command is to assume the two errors are uncorrelated, so the `cov(e.y1*e.y2)` option needs to be added. Otherwise, `sem` would merely provide OLS estimates of

each equation, which would be inconsistent because of simultaneous equations bias.

23.5.6 sem path builder

Stata provides a path builder that enables one to draw a path analysis diagram. Given this path diagram, it automatically generates the `sem` command to fit the model.

We provide two examples of path diagrams in the following analysis. Many examples are given in the *Structural Equation Modeling Reference Manual*. The interactive SEM Builder can be initiated in Stata from **Statistics > SEM (structural equation modeling) > Model building and estimation**.

23.5.7 Measurement-error example

For econometrics, the `sem` and `gsem` commands are especially useful for estimation of models with latent variables.

To illustrate this, we consider a case where a regressor is measured with classical measurement error and we have several independent measures of the unobserved latent variable. We show the inconsistency of OLS and how the `sem` command can be used to obtain consistent estimates. We also show how the several independent measures can be viewed as being generated by a common latent variable.

The true DGP is

$$y_i = 0 + z_i + x_i + u_i$$

The problem is that we observe x with error. Instead, we have four measures of x , denoted x_1, x_2, x_3 , and x_4 . The first measure is generated by a classical measurement-error model as

$$x_{1i} = x_i + \varepsilon_{1i}, \quad \varepsilon_{1i} \sim (0, \sigma_{1i}^2)$$

OLS regression of y on z and x_1 will yield inconsistent estimates of β_z and β_{x_1} because of the measurement-error problem. The other three measures are similarly generated. Note that the measurement errors ε_{1i} , ε_{2i} , ε_{3i} , and ε_{4i} are independent. We have

```
. * SEM measurement error example: Generate the data
. clear
. qui set obs 1000
. set seed 10101
. gen x = rnormal(0,1)
. gen z = x + rnormal(0,1)
. gen y = 1 + 1*x + 1*z + rnormal(0,1)
. gen x1 = x + rnormal(0,1)
. gen x2 = x + rnormal(0,1)
. gen x3 = x + rnormal(0,1)
. gen x4 = x + rnormal(0,1)
```

First, suppose that the true regressor x could be observed, so we can regress y on z and x . Using the `sem` command, we obtain

```

. * SEM measurement error: OLS (using SEM) with true regressor x consistent
. sem y <- z x, nolog vce(robust)

Endogenous variables
    Observed: y

Exogenous variables
    Observed: z x

Structural equation model
Number of obs = 1,000
Estimation method: ml
Log pseudolikelihood = -4277.4193

```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
Structural					
y					
z	1.057966	.0328901	32.17	0.000	.993503 1.12243
x	.9488263	.044741	21.21	0.000	.8611356 1.036517
_cons	1.024673	.0317183	32.31	0.000	.9625066 1.08684
var(e.y)	1.007609	.0451899			.9228197 1.100189

As expected, OLS is consistent, and all estimated coefficients are close to their DGP values of 1.

Now, suppose we use the mismeasured regressor x_1 and regress y on z and x_1 . We obtain

```

. * SEM measurement error: OLS with mismeasured regressor x1 inconsistent
. sem y <- z x1, nolog vce(robust)
Endogenous variables
    Observed: y
Exogenous variables
    Observed: z x1
Structural equation model
Estimation method: ml
Number of obs = 1,000
Log pseudolikelihood = -4919.2122

```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
Structural					
y					
z	1.371866	.0299465	45.81	0.000	1.313172 1.43056
x1	.3380016	.0300409	11.25	0.000	.2791224 .3968807
_cons	1.019155	.0361965	28.16	0.000	.9482112 1.090099
var(e.y)	1.305362	.0569675			1.198349 1.42193

Now OLS leads to inconsistent estimation. Note that the coefficient of the mismeasured regressor x_1 is many standard errors from the DGP values of 1 and that there is a spillover effect to the coefficient of z . The mismeasurement also leads to a poorer model fit because the estimated error variance is now substantially more than 1.

The standard solution is IV regression. For example, we could include x_1 as a regressor and use x_2 , x_3 , and x_4 as instruments. This can be estimated using the command `ivregress 2sls y z (x1 = x2 x3 x4)`.

Using the `sem` command instead, we have

```

. * SEM measurement error: IV with x2, x3, x4 instruments with x1 consistent
. sem (y <- z x1) (x1 <- z x2 x3 x4), covar(e.y*e.x1) nolog noheader vce(robust)
Endogenous variables
    Observed: y x1
Exogenous variables
    Observed: z x2 x3 x4

```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
y					
x1	1.003994	.0881314	11.39	0.000	.8312594 1.176728
z	1.048039	.0508024	20.63	0.000	.9484687 1.14761
_cons	1.044999	.0442263	23.63	0.000	.9583173 1.131681
x1					
z	.2116979	.0303231	6.98	0.000	.1522657 .27113
x2	.2241763	.02473	9.06	0.000	.1757063 .2726463
x3	.1982505	.023787	8.33	0.000	.1516289 .244872
x4	.1320508	.0239545	5.51	0.000	.0851008 .1790008
_cons	-.0277846	.0343133	-0.81	0.418	-.0950374 .0394683
var(e.y)	1.937652	.1829044			1.610374 2.331443
var(e.x1)	1.169616	.0509586			1.073885 1.273882
cov(e.y,e.x1)	-.9493958	.1126678	-8.43	0.000	-1.170221 -.728571

The first equation estimates are close to the DGP values. The standard errors of z and x_1 are 0.0508 and 0.0881 compared with preceding standard errors of z and the true regressor x of 0.0329 and 0.0447. The `sem` command obtains ML estimates. Thus, for this overidentified model, the equivalent `ivregress 2sls` command gives somewhat different estimates, but the `ivregress liml` command gives identical estimates.

23.5.8 Measurement-error model estimates using a latent variable

As an alternative method to obtain consistent estimates, we will perform OLS regression with the mismeasured regressor x_1 replaced by a latent variable x that determines the mismeasured observed variables x_1 , x_2 , and x_3 . This example illustrates the use of latent variables.

SEM for the mismeasured regressor

Before moving to the desired regression, we model the latent variable using a one-factor measurement model. Such models are widely used in psychology, for example, where we might have three noisy measures of ability that are viewed as being generated by a single unobserved latent variable for ability.

Thus, we suppose there is an unobserved or latent variable X underlying each of x_1 , x_2 , and x_3 . That is, $X_i \sim (0, \sigma_X^2)$ and

$$\begin{aligned}x_{1i} &= \alpha_1 + \gamma_1 X_i + \varepsilon_{1i}, & \varepsilon_{1i} &\sim (0, \sigma_1^2) \\x_{2i} &= \alpha_2 + \gamma_2 X_i + \varepsilon_{2i}, & \varepsilon_{2i} &\sim (0, \sigma_2^2) \\x_{3i} &= \alpha_3 + \gamma_3 X_i + \varepsilon_{3i}, & \varepsilon_{3i} &\sim (0, \sigma_3^2)\end{aligned}$$

The latent variable X_i requires a normalization. The `sem` command does this by setting $\gamma_1 = 1$. With this normalization and given the DGP, we expect that, approximately, the intercepts $\alpha_1 = \alpha_2 = \alpha_3 = 0$, the coefficients of the latent variable $\gamma_1 = \gamma_2 = \gamma_3 = 1$, while the three error variances should be equal to 1.

Figure 23.4 presents a path diagram for this model. The observed variables x_1 , x_2 , and x_3 are represented in rectangles, the latent variable x is represented in a circle and is capitalized, and the three errors, also not observed, are represented in circles. The errors are statistically independent, and each determines only one variable. The single latent variable x determines all three of the observed variables.

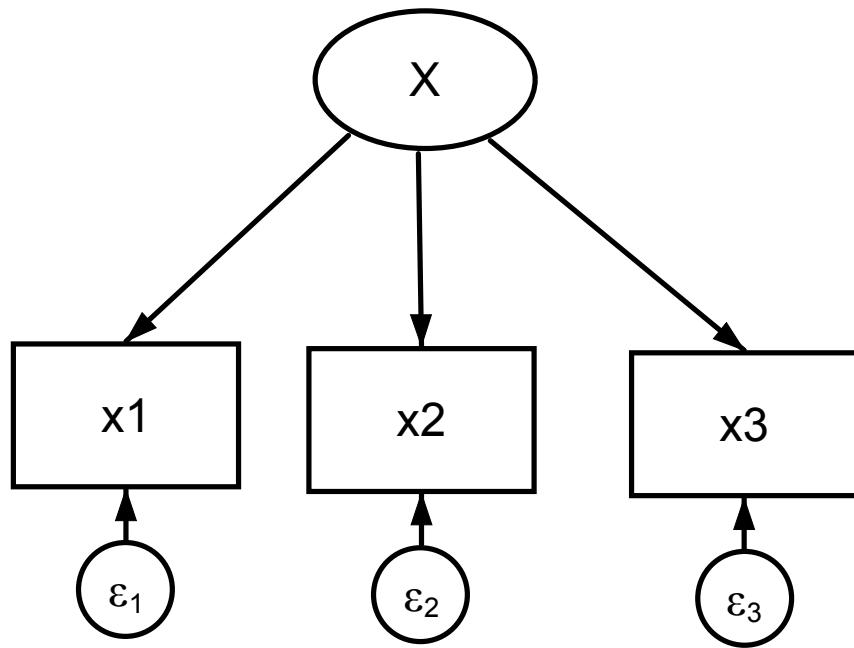


Figure 23.4. Path diagram for measurement-error model

For the `sem` command, the observed variable names must be in lowercase, while uppercase is reserved for any latent variables, in this case `X`. We have

```
. * SEM measurement error: Measurement model for X using 3 measures x1, x2, x3
. sem (x1 x2 x3 <- X), nolog
```

Endogenous variables

Measurement: x1 x2 x3

Exogenous variables

Latent: X

Structural equation model

Number of obs = 1,000

Estimation method: ml

Log likelihood = -4932.9313

(1) [x1]X = 1

	OIM					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
Measurement x1						
	X _cons	1 (constrained) .015836	.0436283	-0.36	0.717	-.1013458 .0696739
x2						
	X _cons	.9067773 .0237131	.0597044 .0429141	15.19 0.55	0.000 0.581	.7897588 1.023796 -.0603971 .1078232
x3						
	X _cons	1.05723 .0208704	.0694522 .0461941	15.22 0.45	0.000 0.651	.9211066 1.193354 -.0696684 .1114093
var(e.x1)	.9230685	.0699544			.7956572	1.070883
var(e.x2)	1.035528	.0655821			.9146467	1.172385
var(e.x3)	1.038118	.078359			.8953573	1.203641
var(X)	.9803578	.0934423			.8133038	1.181725

LR test of model vs. saturated: chi2(0) = 0.00

Prob > chi2 = .

The parameter estimates are close to their expected values of 0 for intercepts and 1 for the remaining parameters. Note the SEM terminology that the measured variables x_1 , x_2 , and x_3 are viewed as endogenous because they are determined by another variable. This other variable, the latent variable X , is viewed as exogenous because it is determined by no variable, only by a model error.

What if we had used a different value normalization of the latent variable? Suppose we constrain the first slope coefficient to be 2 rather than 1. Then $x_{1i} = \alpha_1 + 1 \times X_i + w_{1i}$ becomes $x_{1i} = \alpha_1 + 2 \times (X_i/2) + w_{1i}$, so setting $\gamma_1 = 2$ is equivalent to halving the latent variable. Otherwise,

there is no change. So the only coefficients that should change are γ_2 and γ_3 , which double, and the estimated variance of the latent variable, which is one quarter as large. From output not given, the command `sem (x1 <- x@2) (x2 <- x) (x3 <- x)` confirms that this is indeed the case.

An alternative normalization is to normalize the variance of the latent variable. For example, giving the command `sem (x1 x2 x3 <- x), var(x@0.25)` leads to the three estimated slope coefficients γ_1 , γ_2 , and γ_3 changing by the same multiple, while the estimated intercepts and the estimated error variances are unchanged.

From theory given below, SEM coefficients are identified from the covariance matrix of observed variables. In this example, there are 3 observed variables, so the 3×3 covariance matrix has $3 \times 4/2 = 6$ unique entries. There are nine parameters. Three are identified by the means of the three observed variables `x1`, `x2`, and `x3`. The remaining six are identified by the covariance matrix, so the model is exactly identified. Thus, the LR test at the end of the output cannot be implemented.

What if we used only two of the measures? With only two measures, the covariance matrix has $2 \times 3/2 = 3$ entries, but there are four parameters, and the model is not identified. In fact, the command `sem (x1 x2 <- x)` leads to a warning message that the model is not of full rank. Consequently, the variance of the error for variable `x1` was very small (3.12×10^{-9}), and no confidence intervals were given for this variance or for the variance of the latent variable.

With all four measures used, the covariance matrix has $4 \times 5/2 = 10$ entries and 8 parameters. The model is overidentified, and the LR test has 2 degrees of freedom. From output not given, the $\chi^2(2)$ test statistic has value 6.12 and p -value of 0.047, so at level 0.05, we would just reject the constraints implied by the measurement model. Given the DGP, we do not expect such a rejection, though a rejection will occur in 5% of simulations. If we reset the seed generating the data to 12345, for example, then we find that the test statistic equals 0.65 with $p = 0.724$.

SEM regression controlling for mismeasured regressor

We now move onto estimation of the model for y , with the mismeasured variable x_1 replaced by the latent variable x obtained from the preceding one-factor measurement model. We have

$$y_i = \beta_1 + \beta_{21}z_i + \beta_3X_i + u_i$$

where additionally we have data on x_1 , x_2 , and x_3 that are related to x .

Figure 23.5 presents a path diagram for this model. The observed dependent variable y is determined by the observed variable z and the latent variable x . The latent variable x determines the three observed measurements x_1 , x_2 , and x_3 . The four errors are ε_1 , ε_2 , ε_3 , and u .

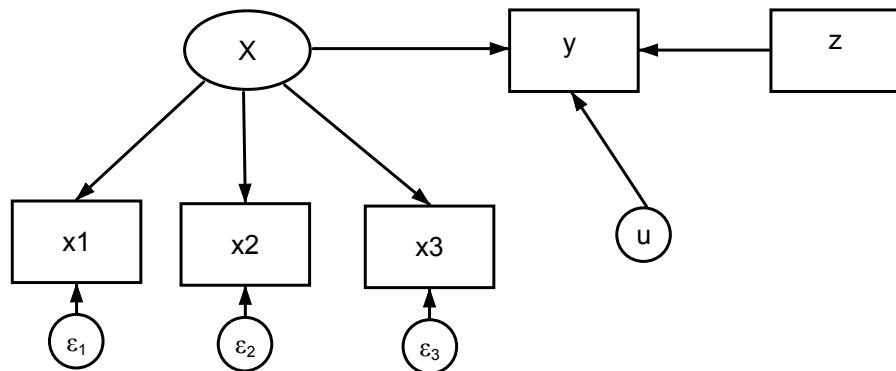


Figure 23.5. Path diagram for regression with measurement error

We have

```
. * SEM meas error: regress y on z and x controlling for measurement error in x
. sem (x1 x2 x3 <- X) (y <- z X), nolog vce(robust)
```

Endogenous variables

Observed: y

Measurement: x1 x2 x3

Exogenous variables

Observed: z

Latent: X

Structural equation model

Number of obs = 1,000

Estimation method: ml

Log pseudolikelihood = -7959.6236

(1) [x1]X = 1

	Robust				
	Coefficient	std. err.	z	P> z	[95% conf. interval]
Structural					
y					
z	1.010318	.0457044	22.11	0.000	.9207387 1.099897
X	1.066612	.087781	12.15	0.000	.8945646 1.23866
_cons	1.030882	.0384079	26.84	0.000	.9556038 1.10616
Measurement					
x1					
X		1 (constrained)			
_cons	-.015836	.0377764	-0.42	0.675	-.0898764 .0582045
x2					
X	.9791609	.0562687	17.40	0.000	.8688763 1.089446
_cons	.0237131	.0369525	0.64	0.521	-.0487124 .0961386
x3					
X	1.043577	.0540005	19.33	0.000	.9377379 1.149416
_cons	.0208704	.0399025	0.52	0.601	-.057337 .0990779
var(e.x1)	.9634506	.0520942			.8665719 1.07116
var(e.x2)	.9404119	.0527446			.8425138 1.049686
var(e.x3)	1.110213	.0647267			.9903305 1.244607
var(e.y)	.9578786	.0570865			.8522786 1.076563
var(X)	.9399771	.0794081			.7965423 1.10924
cov(z,X)	.9966285	.0489429	20.36	0.000	.9007021 1.092555

All coefficients are close to their DGP values. As expected, there is an efficiency loss compared with the case where the true regressor x is instead included in the regression; the standard errors of z and x are 0.0457 and 0.0878 compared with standard errors of z and x of 0.0329 and 0.0447.

There are 5 observed variables, so there are $5 \times 6/2 = 15$ covariances. These covariances are used to identify 11 parameters (the remaining parameters are identified by the means of the 5 observed variables). The overidentifying restrictions test, available if the `vce(robust)` option is dropped, has $15 - 11 = 4$ degrees of freedom and does not reject the restrictions of the model at level 0.05 because $p = 0.096 > 0.05$. Note that we have assumed that the three measurement errors are independent, an important restriction on error covariances that ensures identification of the model.

How do we interpret the model coefficients? Clearly, a one-unit change in the latent variable x is associated with a 1.067-unit change in y . But what does a one-unit change in x mean? One approach is to note that a one-unit change in x_1 , x_2 , or x_3 is associated with an approximate one-unit change in x . So we might view a one-unit change in the latent variable x as the same as a one-unit change in the observed variable x . Then a one-unit change in x is viewed as associated with a 1.067-unit change in y , after controlling for measurement error.

Finally, we note in passing that the extent of measurement error is measured by the reliability ratio—the variance of the true variable divided by the variance of the mismeasured variable. If this is known, as it is in this example with known DGP, then one can use the `reliability()` option of the `sem` command.

23.5.9 SEM endogenous regressors example

As an example, we consider a just-identified two-equation model, using the same DGP as in section 7.2. Then $y_1 = \beta_{12}y_2 + \gamma_{12}x_1 + u_1$, $y_2 = \beta_{21}y_1 + \gamma_{21}x_2 + u_2$, and the errors u_1 and u_2 are correlated. From section 7.2, $\beta_{12} = \gamma_{12} = \gamma_{21} = 1$ and $\beta_{21} = 0.25$.

```
. * DGP for just-identified two-equation simultaneous equations example
. clear all
. set seed 10101
. drawnorm u1 u2, n(1000) corr(1, .7, 1) cstorage(lower)
(obs 1,000)
. drawnorm x1 x2, n(1000) corr(1, .3, 1) cstorage(lower)
```

```
. 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
```

We initially focus on estimation of just the y_1 equation. The `sem` command equivalent to `ivregress` uses the structural form for y_1 and the reduced form for y_2 . For joint estimation of the y_1 and y_2 equations, the `sem` command equivalent to `reg3` uses both structural form equations.

```
. * SEM commands for 2SLS and 3SLS in just-identified model  
. qui ivregress 2sls y1 (y2=x2) x1, noheader  
. estimates store iv_2sls  
. qui sem (y1 <- y2 x1) (y2 <- x1 x2), covar(e.y1*e.y2)  
. estimates store sem_rf  
. qui reg3 (y1 = y2 x1) (y2 = y1 x2)  
. estimates store iv_3sls  
. qui sem (y1 <- y2 x1) (y2 <- y1 x2), covar(e.y1*e.y2)  
. estimates store sem_sf
```

We next compare the various estimates of the parameters of the y_1 equation.

```
. * SEM estimators compared with 2SLS and 3SLS in just-identified model
. estimates table iv_2sls sem_rf iv_3sls sem_sf, eq(1) b(%9.5f) se
```

Variable	iv_2sls	sem_rf	iv_3sls	sem_sf
#1				
y2	0.95510 0.02797	0.95510 0.02797	0.95510 0.02797	0.95510 0.02797
x1	1.04403 0.04095	1.04403 0.04095	1.04403 0.04095	1.04403 0.04095
_cons	0.00481 0.03383	0.00481 0.03383	0.00481 0.03383	0.00481 0.03383
y2				
x1		0.41523 0.05307		
x2		1.27681 0.05322	0.92532 0.04617	0.92532 0.04617
y1			0.28823 0.02121	0.28823 0.02121
_cons		0.04345 0.05040	0.03010 0.02907	0.03010 0.02907
var(e.y1)		1.14308 0.09368		1.14308 0.09368
var(e.y2)		2.53733 0.11347		0.84140 0.09505
cov(e.y1, e.y2)		1.40313 0.09954		0.68740 0.06958

Legend: b/se

The estimates for the y_1 equation are equivalent because in a just-identified system 2SLS, limited information ML, 3SLS, and full information ML are equivalent. In the overidentified case, there will be finite sample differences because SEM estimators are ML estimators, while 2SLS and 3SLS estimators are moment-based estimators.

23.5.10 SEM estimation in general

The linear SEM approach in general specifies a linear model involving measured variables, such as y_i and x_i , and latent or unobserved variables,

such as model errors. As detailed below, this accommodates a very wide range of models that are routinely used in empirical social science research.

Let \mathbf{z}_i denote the measured variables, and let $\boldsymbol{\theta}$ denote all the parameters of the model. Without any structure, the sample mean vector and variance matrix are

$$\bar{\mathbf{z}} = \frac{1}{N-1} \sum_{i=1}^N \mathbf{z}_i$$

$$\mathbf{S}_{zz} = \frac{1}{N-1} \sum_{i=1}^N (\mathbf{z}_i - \bar{\mathbf{z}})(\mathbf{z}_i - \bar{\mathbf{z}})'$$

A SEM, like the preceding linear regression example, implies that

$$\begin{bmatrix} E(\mathbf{z}_i) \\ \text{Var}(\mathbf{z}_i) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}(\boldsymbol{\theta}) \\ \boldsymbol{\Sigma}(\boldsymbol{\theta}) \end{bmatrix}$$

where different SEMs lead to different functional forms for $\boldsymbol{\mu}(\boldsymbol{\theta})$ and $\boldsymbol{\Sigma}(\boldsymbol{\theta})$.

This model can be fit in two ways.

ML estimation

Under the assumption of normality of \mathbf{z}_i and independence over i , the log-likelihood function has the same form as that in (23.5). This can be shown to simplify to

$$\ln L(\boldsymbol{\theta}) = -\frac{N}{2} \left[K \ln 2\pi + \ln\{\det \boldsymbol{\Sigma}(\boldsymbol{\theta})\} + \text{trace} \left\{ \mathbf{D} \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1} \right\} \right] \quad (23.6)$$

$$\mathbf{D} = \mathbf{S}_{zz} + \{\bar{\mathbf{z}} - \boldsymbol{\mu}(\boldsymbol{\theta})\}\{\bar{\mathbf{z}} - \boldsymbol{\mu}(\boldsymbol{\theta})\}'$$

Strictly speaking, $\mathbf{D} = \mathbf{S}_{zz}^* + \{\bar{\mathbf{z}} - \boldsymbol{\mu}(\boldsymbol{\theta})\}\{\bar{\mathbf{z}} - \boldsymbol{\mu}(\boldsymbol{\theta})\}'$, where \mathbf{S}_{zz}^* divides the sum by N rather than $N-1$. Some other SEM packages normalize by N , leading to slightly different results.

This estimator is obtained using `sem` command option `method(ml)`, the default. The estimates are robust to nonnormality in the model errors, though not to nonnormality in the latent variables because this leads to misspecification of $\Sigma(\theta)$. Robust standard errors can be obtained using the `vce(robust)` option. A wide range of `vce()` options are available, including `cluster` and `sbentler`.

The `method(mlmv)` option uses imputation methods under the assumptions of missingness at random and joint normality of all variables (both observed and latent) when data include missing values. This leads to less loss of precision than if casewise deletion is used, at the expense of stronger distributional assumptions.

Moment-based estimation

An alternative approach matches sample moments and model moments, so that $\bar{\mathbf{z}} = \mu(\hat{\theta})$ and $\mathbf{S}_{zz} = \Sigma(\hat{\theta})$. The preceding linear regression model example was just identified, and there was a unique solution for θ . More generally, SEMs may be overidentified, in which case there is no unique solution. Then estimation is by generalized method of moments (GMM). We stack the sample mean vector and sample variance matrix, similarly stack the corresponding parameter restrictions, and define

$$\mathbf{w} = \begin{bmatrix} \bar{\mathbf{z}} \\ \text{vech}(\mathbf{S}_{zz}) \end{bmatrix} \text{ and } \boldsymbol{\tau}(\theta) = \begin{bmatrix} \mu(\theta) \\ \text{vech}\{\Sigma(\theta)\} \end{bmatrix}$$

where `vech(.)` is the vector-half operator that vectorizes the lower triangular half of a symmetric matrix. In the just-identified case, we could solve $\mathbf{w} = \boldsymbol{\tau}(\theta)$ for θ .

In the overidentified case, we solve a quadratic form in the discrepancy $\{\mathbf{w} - \boldsymbol{\tau}(\theta)\}$. Then $\hat{\theta}$ minimizes

$$Q(\theta) = \{\mathbf{w} - \boldsymbol{\tau}(\theta)\}' \mathbf{W}^{-1} \{\mathbf{w} - \boldsymbol{\tau}(\theta)\} \quad (23.7)$$

where \mathbf{W}^{-1} is a weighting matrix. This GMM estimator is often called weighted least squares in the SEM literature and is also called a minimum chi-squared estimator. If the model is just identified, all choices of \mathbf{W} lead to the same estimate $\hat{\boldsymbol{\theta}}$ and $Q(\hat{\boldsymbol{\theta}}) = 0$. If the model is overidentified and \mathbf{W}^{-1} is the optimal weighting matrix, then the departure of $Q(\hat{\boldsymbol{\theta}})$ from 0 provides a test of overidentifying restrictions.

This estimator is also called the asymptotic distribution-free estimator and is implemented using the `method(adf)` option of the `sem` command, which specifies \mathbf{W} to be the estimated covariance matrix of \mathbf{w} . This weighting matrix requires estimating the variances and covariance of the components of $\bar{\mathbf{z}}$ and \mathbf{S}_{zz} .

Consistency of this estimator requires that $E\{\mathbf{w} - \boldsymbol{\tau}(\boldsymbol{\mu}, \boldsymbol{\theta})\} = \mathbf{0}$, so the variance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ needs to be correctly specified. The weighting matrix used is the optimal weighting matrix for GMM based on the moment condition $E\{\mathbf{w} - \boldsymbol{\tau}(\boldsymbol{\mu}, \boldsymbol{\theta})\} = \mathbf{0}$ and assuming independence over i . So the default standard errors are automatically robust, provided the observations are independent. If observations are instead clustered, but we still assume that $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ is correctly specified, then cluster-robust standard errors can be obtained by using a cluster bootstrap, provided the number of clusters is sufficiently large.

Standard errors and overidentifying restrictions test

A range of standard errors can be obtained after `method(ml)`, including heteroskedastic-robust and cluster-robust standard errors. However, the overidentifying restrictions test is possible only when default standard errors are used (option `vce(oim)`) or when heteroskedastic-robust standard errors are obtained using option `vce(sbentler)`. After `method(adf)`, the default standard errors (option `vce(oim)`) are essentially heteroskedastic robust. Cluster-robust standard errors are then not available, though bootstrap and jackknife standard errors are.

Computational considerations

With K observed variables, there are $K(K + 1)/2$ variance components, so at most there can be $K(K + 1)/2$ parameters, excluding the intercepts and

means. One sign of nonidentification is that the log likelihood is not changing over iterations. In that case, one can set the number of iterations to the number where this problem arises, using the `iterate(#)` option. Parameter estimates in the resulting output that have missing standard errors are signals that the parameter is unidentified.

Numerical problems may arise. Even though the SEM is linear in parameters, the implied covariance matrix $\Sigma(\boldsymbol{\theta})$ can be nonlinear in parameters, leading to numerical problems in maximizing $\ln L(\boldsymbol{\theta})$ or minimizing $Q(\boldsymbol{\theta})$.

From (23.6), the ML methods just require data on \bar{z} and S_{zz} , rather than on individual observations. One can just provide these rather than all the data. The `ssd` command creates a special dataset containing the values of \bar{z} and S_{zz} that can be used to fit models with `sem`.

Normalization

Normalization constraints on the latent variables are necessary because these, unlike observed variables, are not naturally scaled. The model $y = 100 + \eta$ with $\eta \sim (100, 25)$ is equivalent to the model $y = 200 + 5\eta$ with $\eta \sim (0, 1)$.

To normalize the centering, Stata sets latent exogenous variables to have mean 0 and latent endogenous variables to have intercept 0. To set the scale, Stata sets the coefficients on paths from latent variables to the first observed endogenous variable to 1 and, if there is no endogenous variable, sets the coefficients on paths from latent variables to the first endogenous latent variable to 1 if the latent variable is measured by other latent variables only.

23.6 Generalized structural equation models

The `mixed` command is relevant for multilevel linear models with latent normal variables for a single dependent variable. The `me` commands extend the `mixed` command to multilevel GLMS with latent normal variables for a single dependent variable. The `sem` command extends the `mixed` command to permit more than one dependent variable but is restricted to a single level.

The `gsem` command extends the `sem` command to GLMS, presented in section 13.3.7. Furthermore, it allows for multiple levels. Simultaneity is permitted, but, aside from linear models, the models must be recursive.

The `gsem` command provides several features not provided by the `sem` command. It enables one to

1. fit SEMs containing generalized linear response variables;
2. fit recursive simultaneous equation nonlinear SEMS;
3. fit SEMs with categorical latent variables (FMMS);
4. use factor-variable notation (not possible with the `sem` command); and
5. fit multilevel mixed SEMS.

We focus on the use of `gsem` commands for single-level models and their use in fitting more flexible parametric models with unobserved heterogeneity, such as for models with selection on unobservables. An example recursive two-equation model with one level specifies

$$\begin{aligned} E(y_{1i}|\mathbf{x}_{1i}, y_{2i}, u_i) &= g_1^{-1}(\alpha y_{2i} + \mathbf{x}'_{1i}\boldsymbol{\beta}_1 + \delta u_i), & y_{1i} \sim F_1 \\ E(y_{2i}|\mathbf{x}_{2i}, u_i) &= g_2^{-1}(\mathbf{x}'_{2i}\boldsymbol{\beta} + u_i), & y_{2i} \sim F_2 \\ u_i &\sim N(0, \sigma_u^2) \end{aligned}$$

where $g_1(\cdot)$ and $g_2(\cdot)$ are link functions and F_1 and F_2 are densities for standard GLMS. The model is recursive because a model for $y_1|y_2$ is specified, but the model specified for y_2 does not depend on y_1 .

The `gsem` command applies to many models and data types: the normal for unbounded data; logit, probit, and complementary log–log for binary data; multinomial, ordered logit, ordered probit, and ordered complementary log–log for multinomial data; beta for continuous data on $(0, 1)$; lognormal,

exponential, gamma, loglogistic, and Weibull for continuous positive data; and Poisson and negative binomial for count data.

23.6.1 GSEM estimation

Models with continuous latent variables are fit by ML. Let \mathbf{y} denote the vector of dependent variables, \mathbf{X} the regressor matrix, and \mathbf{u} the vector of latent variables. If \mathbf{u} was observed, the likelihood would be the joint density of the data $f(\mathbf{y}|\mathbf{X}, \mathbf{u}, \boldsymbol{\theta})$. Instead, \mathbf{u} is an unobserved normally distributed variable that needs to be integrated out. So the likelihood function is

$$L(\boldsymbol{\theta}) = \int f(\mathbf{y}|\mathbf{X}, \mathbf{u}, \boldsymbol{\theta})\phi(\mathbf{u}|\boldsymbol{\mu}_{\mathbf{u}}, \boldsymbol{\Sigma}_{\mathbf{u}})d\mathbf{u}$$

where $\phi(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the density of the multivariate normal with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$.

This integral has no closed-form solution, aside from exceptions such as linear SEMs, and is instead computed numerically using quadrature methods. And, as for the `me` commands, any model misspecification will lead to inconsistent parameter estimates.

23.6.2 The gsem command

The `gsem` command, like the `sem` command, is very complicated. We provide a very brief summary. The syntax is

```
gsem paths [if] [in] [weight] [, options]
```

where the *path* defines the GSEM in command-language path notation; examples are given below.

The `family()` and `link()` options define the particular GLM used. The options `covariance()`, `variance()`, and `mean()` override the defaults for the latent variables. The `lclass()` option defines latent classes.

The `gsem` command provides four numerical integration methods via the option `intmethod(): mvaghermite` (the default), `mcaghermite`, `ghermite`, and `laplace`. The methods were described in more detail in section [23.4.1](#).

23.6.3 Some GSEM examples

Table [23.2](#) presents some example `gsem` commands. The first five examples are identical to some leading standard Stata estimation commands and yield exactly the same results. The last two entries present two recursive structural model examples that cannot be fit using other commands. The first of these examples has independent errors, while the second introduces error correlation by the latent variable `L`. Uppercase is reserved for latent variable names, while observed variables must be in lowercase.

Table 23.2. `gsem` commands and equivalent standard Stata commands

Estimator	Usual command and SEM equivalent
Logit	<code>logit y x1 x2</code> <code>gsem y <- x1 x2, logit</code> <code>gsem y <- x1 x2, family(binomial) link(logit)</code>
Multinomial logit	<code>mlogit y x1 x2, baseoutcome(1)</code> <code>gsem y <- x1 x2, mlogit</code>
Ordered logit	<code>ologit y x1 x2</code> <code>gsem y <- x1 x2, ologit</code>
Poisson regression	<code>poisson y x1 x2</code> <code>gsem y <- x1 x2, poisson</code>
Linear mixed model	<code>mixed y x id:</code> <code>gsem (y <- x M1[id])</code>
Finite mixture: two count outcomes	<code>gsem (y1 y2 <- x1 x2), poisson lclass(C 2)</code>
Recursive exogenous: binary and ordinal	<code>gsem (y1 <- x1 x2, logit)</code> <code>(y2 <- z1 z2, ologit) (y2 <- y1)</code>
Recursive endogenous: count and continuous	<code>gsem (y1 <- y2 x1 L, poisson)</code> <code>(y2 <- x2 L)</code>

Note that the `gsem` command permits factor-variable notation, such as `c.x1##c.x2`. While recursive models, such as the last two models in the table, can be fit, `gsem` does not cover more general simultaneous equations with y_1 depending on y_2 and y_2 depending on y_1 unless the model is a linear model.

23.6.4 Poisson with latent normal variable for overdispersion

For count-data regression, the Poisson distribution is too restrictive. The negative binomial can be obtained as a Poisson-gamma mixture. An alternative model that accommodates overdispersion is a Poisson-normal mixture that, unlike the negative binomial, has no closed-form solution.

The Poisson-normal mixture model specifies that y_i conditional on regressors \mathbf{x}_i and an unobserved heterogeneity term u_i are Poisson distributed with mean $\exp(\mathbf{x}'_i \boldsymbol{\beta} + u_i)$, where u_i is $N(0, \sigma_u^2)$.

This model can be fit by the `gsem` command with model option `poisson` and introducing a latent variable, denoted here by \mathbb{L} , that is normally distributed with mean 0, coefficient normalized to equal 1, and variance to be estimated. For the doctor-visits example given in section [20.5.4](#), we obtain

```

. * GSEM: Poisson normal mixture model
. qui use mus220mepsdocvis, clear
. global xlist2 medicaid age age2 educyr actlim totchr
. gsem (docvis <- private $xlist2 L, poisson), nolog vce(robust)
Generalized structural equation model                               Number of obs = 3,677
Response: docvis
Family:    Poisson
Link:      Log
Log pseudolikelihood = -10564.921
( 1) [docvis]L = 1

```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
docvis					
private	.1835492	.035085	5.23	0.000	.1147838 .2523146
medicaid	.085144	.0479742	1.77	0.076	-.0088837 .1791716
age	.322203	.061233	5.26	0.000	.2021886 .4422174
age2	-.0021039	.0004067	-5.17	0.000	-.002901 -.0013068
educyr	.0296122	.0045635	6.49	0.000	.0206679 .0385565
actlim	.1647751	.0362693	4.54	0.000	.0936886 .2358615
totchr	.311443	.0123445	25.23	0.000	.2872483 .3356377
L	1 (constrained)				
_cons	-11.78405	2.29393	-5.14	0.000	-16.28007 -7.288027
var(L)	.6272045	.0225134			.5845954 .6729193

The variance is clearly different from zero.

This example is unusual because the conditional mean is unchanged by adding a mixture. We have

$$E(y_i | \mathbf{x}_i, L_i) = \exp(\mathbf{x}'_i \boldsymbol{\beta} + L_i) = \exp(L_i) \times \exp(\mathbf{x}'_i \boldsymbol{\beta}), \text{ so}$$

$E(y_i | \mathbf{x}_i) = a \times \exp(\mathbf{x}'_i \boldsymbol{\beta}) = \exp(\ln a + \mathbf{x}'_i \boldsymbol{\beta})$, where $a = E(L_i)$ simply changes the intercept of the model. This special result holds for a conditional mean that is exponential or linear.

So, provided models are correctly specified, we can compare the slope coefficients across models. We do so for Poisson, Poisson-normal, and negative binomial models.

```
. * GSEM: Compute Poisson, Poisson-normal, and negative binomial
. qui poisson docvis private $xlist2
. estimates store pois_def
. qui poisson docvis private $xlist2, vce(robust)
. estimates store pois_rob
. qui gsem (docvis <- private $xlist2 L, poisson)
. estimates store normal
. qui gsem (docvis <- private $xlist2 L, poisson), vce(robust)
. estimates store norm_rob
. qui nbreg docvis private $xlist2
. estimates store nb2_def
. qui nbreg docvis private $xlist2, vce(robust)
. estimates store nb2_rob
```

We obtain the following estimates, along with default and heteroskedastic–robust standard errors.

```
. * GSEM: Compare Poisson-normal with Poisson and negative binomial
. estimates table pois_def pois_rob normal norm_rob nb2_def nb2_rob,
> eq(1) b(%8.4f) se stfmt(%8.1f) stats(l1 N)
```

Variable	pois_def	pois_rob	normal	norm_rob	nb2_def	nb2_rob
#1						
private	0.1422 0.0143	0.1422 0.0364	0.1835 0.0344	0.1835 0.0351	0.1641 0.0332	0.1641 0.0369
medicaid	0.0970 0.0189	0.0970 0.0568	0.0851 0.0469	0.0851 0.0480	0.1003 0.0454	0.1003 0.0567
age	0.2937 0.0260	0.2937 0.0630	0.3222 0.0618	0.3222 0.0612	0.2941 0.0602	0.2941 0.0646
age2	-0.0019 0.0002	-0.0019 0.0004	-0.0021 0.0004	-0.0021 0.0004	-0.0019 0.0004	-0.0019 0.0004
educyr	0.0296 0.0019	0.0296 0.0048	0.0296 0.0044	0.0296 0.0046	0.0287 0.0042	0.0287 0.0049
actlim	0.1864 0.0146	0.1864 0.0397	0.1648 0.0362	0.1648 0.0363	0.1895 0.0348	0.1895 0.0394
totchr	0.2484 0.0046	0.2484 0.0126	0.3114 0.0122	0.3114 0.0123	0.2776 0.0121	0.2776 0.0132
L			1.0000 0.0000	1.0000 0.0000		
_cons	-10.1822 0.9720	-10.1822 2.3692	-11.7840 2.3128	-11.7840 2.2939	-10.2975 2.2474	-10.2975 2.4241
var(L)			0.6272 0.0221	0.6272 0.0225		
/lnalpha					-0.4453 0.0307	-0.4453 0.0378
Statistics						
l1	-15019.6	-15019.6	-10564.9	-10564.9	-10589.3	-10589.3
N	3677	3677	3677	3677	3677	3677

Legend: b/se

The slope coefficients vary by 10% to 30% across the models. The Poisson-normal model fits better than the negative binomial with a log likelihood that is 25.6 higher, and with robust standard errors that are quite close to default standard errors because the normally distributed unobserved heterogeneity term controls for overdispersion.

23.6.5 Poisson model with endogeneity

We consider the structural-model approach of section [20.7.2](#), a model for `docvis` with endogenous regressor `private`. There a two-step estimator was used. Here we consider a more structural model that introduces an error term u

that is common to both the Poisson equation of interest and to the first-stage equation for the endogenous regressor.

To do so, we obtain the MLE using the `gsem` command. For both equations, we add the latent variable `L`. We obtain

```
. * GSEM: Endogenous regressor in a Poisson model
. gsem (docvis <- private $xlist2 L, poisson)
>      (private <- $xlist2 income ssiratio L), nolog vce(robust)
Generalized structural equation model                                         Number of obs = 3,677
Response: docvis
Family:  Poisson
Link:    Log
Response: private
Family:  Gaussian
Link:    Identity
Log pseudolikelihood = -12796.769
( 1) [docvis]L = 1
```

	Coefficient	Robust std. err.	z	P> z	[95% conf. interval]
docvis					
private	.6331611	.2194449	2.89	0.004	.2030571 1.063265
medicaid	.2675884	.1005366	2.66	0.008	.0705402 .4646366
age	.3687153	.0660348	5.58	0.000	.2392895 .4981411
age2	-.0023977	.0004366	-5.49	0.000	-.0032533 -.001542
educyr	.0176344	.0073831	2.39	0.017	.0031637 .0321051
actlim	.1852541	.0385927	4.80	0.000	.1096138 .2608945
totchr	.3038513	.0130789	23.23	0.000	.2782171 .3294855
L	1	(constrained)			
_cons	-13.71873	2.504824	-5.48	0.000	-18.6281 -8.809367
private					
medicaid	-.3945375	.0173646	-22.72	0.000	-.4285714 -.3605036
age	-.0829381	.0293402	-2.83	0.005	-.1404439 -.0254323
age2	.0005247	.0001956	2.68	0.007	.0001412 .0009081
educyr	.0213581	.0020524	10.41	0.000	.0173355 .0253806
actlim	-.0297923	.0176555	-1.69	0.092	-.0643966 .0048119
totchr	.0185451	.0057356	3.23	0.001	.0073036 .0297865
income	.0026301	.0004877	5.39	0.000	.0016742 .0035859
ssiratio	-.0724733	.0211669	-3.42	0.001	-.1139597 -.030987
L	-.1355801	.0575232	-2.36	0.018	-.2483236 -.0228367
_cons	3.528705	1.094523	3.22	0.001	1.38348 5.673931
var(L)	.6679536	.047179		.5815997	.7671291
var(e.private)	.1850348	.0110503		.164596	.2080115

The structural equation coefficient estimates are generally within 20% of the two-step estimates given in section [20.7.2](#).

The latent variable is statistically significant at 5%. In this example, the Stata command normalizes the coefficient of the latent variable L to equal one in the linear model.

23.6.6 Prediction and MEs

For GSEMs that include latent variables, predictions and MEs vary with the method used to control for the latent variables. The methods used are similar to those for nonlinear mixed models detailed in section [23.4.3](#).

For `predict`, there are four options `mu conditional(ebmeans)`, the default; `mu conditional(ebmodes)`; `mu conditional(fixedonly)`; and `mu marginal`, which numerically integrates out the latent variables.

We apply several of these prediction options to the current example.

```
. * GSEM: Predict y controlling for latent variable L
. predict mu_ebmeans, mu conditional(ebmeans)
(using 7 quadrature points)
. predict mu_fixed, mu conditional(fixedonly)
note: The prediction for `docvis' has an observed endogenous regressor.
note: Observed endogenous regressors will be treated as fixed at their current
values.
. predict mu_marg, mu marginal
note: The prediction for `docvis' has an observed endogenous regressor.
note: Observed endogenous regressors will be treated as fixed at their current
values.
```

The notes in the output indicate that the predictions are based on the observed values of the endogenous regressor `private`, which is fine because the predictions use parameter estimates that control for this endogeneity.

The predictions and their correlations are as follows.

```
. * GSEM: Compare predictions of y
. summarize docvis mu_ebmeans mu_fixed mu_marg
```

Variable	Obs	Mean	Std. dev.	Min	Max
docvis	3,677	6.822682	7.394937	0	144
mu_ebmeans	3,677	6.469881	6.545503	.7709767	138.1847
mu_fixed	3,677	5.361157	3.357089	1.165272	32.09691
mu_marg	3,677	7.486914	4.688211	1.627315	44.82368

```
. correlate docvis mu_ebmeans mu_fixed mu_marg
(obs=3,677)
```

	docvis mu_ebm~s mu_fixed mu_marg			
docvis	1.0000			
mu_ebmeans	0.9942	1.0000		
mu_fixed	0.3567	0.4036	1.0000	
mu_marg	0.3567	0.4036	1.0000	1.0000

The `conditional(ebmeans)` option estimates a mean for each grouping of observations, but here there is no grouping, so it is close to including a separate fixed effect for each individual. Thus, the prediction `mu_ebmeans` essentially equals the sample value of `docvis`. The `fixedonly` prediction sets $L = 0$ for each observation, so the fitted value for y_2 is simply $\exp(\hat{\beta}_1 y_2 + \mathbf{x}'_1 \hat{\beta}_2)$. Because the conditional mean here is of exponential form, the `marginal` prediction is a simple multiple of this; see section [23.6.4](#) for an explanation. So `mu_fixed` and `mu_marg` are perfectly correlated.

The `margins` postestimation command is not applicable to the current example, because the command works only for statistics whose computation does not involve latent variables. Instead, MES can be obtained using the manual method of section 13.7.10. First, predict using the `mu marginal` option. Second, compute a numerical derivative by perturbing the regressor by a small amount, repredicting, and comparing average predictions. Third, embed this in a bootstrap loop to get a standard error for the AME.

23.6.7 GSEM for FMM with multiple outcomes

The `fmm` prefix, presented in section [23.2](#), estimates single-equation FMMS. These models can also be fit using the `gsem` command. For example, exactly the same estimates are obtained by the commands `fmm 2: poisson y x1 x2` and `gsem (y <- x1 x2), poisson lclass(C 2)`.

The `gsem` command can be extended to allow estimation of an FMM with more than one outcome variable. Examples include pairs of binary, multinomial, count, or continuous outcomes. The simultaneous estimation of multiple FMMs provides combined output that facilitates comparison across responses. Additionally, it is easy to impose and test parametric restrictions across equations. These advantages are appealing especially when the multiple outcomes are related or similar in some way.

We use data on the number of annual emergency room visits (`emr`) and hospitalizations (`hosp`) from the Medicare population. The regressors in the model are `privins` (private insurance), `medicaid` (public insurance), `numchron` (number of chronic conditions), `age` (age/10), and `male`. The outcome variables are positively correlated. Because these outcomes are relatively rare even in the elderly population, the outcomes have a high proportion of zero events.

```
. * GSEM: Read in, and describe data for bivariate dependent variables example
. qui use mus223gsemfmmexample, clear
. global xlist privins medicaid numchron age male
. summarize emr hosp $xlist
```

Variable	Obs	Mean	Std. dev.	Min	Max
emr	4,355	.2597015	.6986564	0	12
hosp	4,355	.289093	.7342015	0	8
privins	4,355	.7752009	.4174979	0	1
medicaid	4,355	.0911596	.287869	0	1
numchron	4,355	1.534099	1.346467	0	8
age	4,355	7.405052	.6345418	6.6	10.9
male	4,355	.4043628	.4908247	0	1

From output not listed, the correlation of `emr` and `hosp` is 0.477.

We fit a bivariate two-component FMM negative binomial model for the two count outcomes.

```

. * GSEM: Fit two-component FMM bivariate negative binomial
. qui gsem (emr hosp <- $xlist), nbreg lclass(C 2)
>     startvalues(randomid, draws(5) seed(15)) vce(robust)
. estat lcprob

```

Latent class marginal probabilities Number of obs = 4,355

		Delta-method		
		Margin	std. err.	[95% conf. interval]
C	1	.7469986	.026645	.6913295 .7955961
	2	.2530014	.026645	.2044039 .3086705

For brevity, the estimation results from the `gsem` command are suppressed. The complete output yields in order a lengthy log file of over 100 iterations; the log likelihood; parameter estimates for the latent class probabilities; parameter estimates for the first latent class (in order for `emr` and then `hosp`); and parameter estimates for the second latent class results. The latent class probabilities for `emr` and `hosp` are 0.75 and 0.25.

We then obtain the marginal means in the two classes.

```

. * GSEM: Latent class means for two-component FMM bivariate negative binomial
. estat lcmean

```

Latent class marginal means Number of obs = 4,355

		Delta-method				
		Margin	std. err.	z	P> z	[95% conf. interval]
1	emr	.0550896	.0117289	4.70	0.000	.0321014 .0780778
	hosp	.0741005	.0128806	5.75	0.000	.0488551 .099346
2	emr	.8513358	.0637465	13.36	0.000	.7263949 .9762767
	hosp	.9161608	.0683449	13.40	0.000	.7822072 1.050114

The second latent class (with probability 0.25) has mean usage rates of 0.851 and 0.916 for the two outcomes, which are more than 10 times the rates of 0.055 and 0.074 for the first latent class.

As an example of an ME, we compute the AME of the number of chronic conditions.

```

. * GSEM: MEs of numchron
. margins, dydx(numchron)
Average marginal effects                                         Number of obs = 4,355
Model VCE: Robust
dy/dx wrt: numchron
1._predict: Predicted mean (Number of annual emergency room visits), using class
probabiliti, predict(mu outcome(emr))
2._predict: Predicted mean (Number of hospital admissions), using class
probabilities, predict(mu outcome(hosp))

```

		Delta-method				
		dy/dx	std. err.	z	P> z	[95% conf. interval]
numchron -predict	1	.0760105	.008604	8.83	0.000	.059147 .092874
	2	.0953072	.0087897	10.84	0.000	.0780797 .1125347

Both outcomes increase considerably with an additional chronic condition, and the effect is highly statistically significant. Emergency room visits increase by 0.076 compared with a sample average of 0.260, and hospital admissions increase by 0.095 compared with a sample average of 0.289.

23.7 ERM commands for endogeneity and selection

The ERM commands are designed to deal with linear and some nonlinear regression models that include one or more endogenous regressors and, moreover, are subject to potential selectivity bias because of sample selection.

The commands provide ML estimates under the assumption of joint normally distributed errors, and with few exceptions, consistent parameter estimation will require completely correct model specification. The ERM commands use numerical integration methods `intmethod(mvaghermite)`, the default, or `intmethod(ghermite)`; see section [23.4.1](#).

23.7.1 Overview of ERM commands

Table [23.3](#) provides a summary of the ERM commands for cross-sectional data and the complications that they deal with. The table also includes the corresponding `xt` commands, which provide extensions to panel data and clustered data by introducing a random effect to allow for intracluster correlation.

Table 23.3. Summary of ERM commands and command options

Stata ERM command	Outcome model and type of outcome
<code>eregress, xteregress</code>	linear for continuous outcome
<code>eprobit, xtepoprobit</code>	probit for binary outcome
<code>eoprobit, xteoprobit</code>	ordered probit for ordered discrete outcome
<code>eintreg, xteintreg</code>	interval regression for interval data
Stata ERM command option	Complication considered
<code>endogenous()</code>	endogenous continuous regressors
<code>select()</code>	sample selection
<code>tobitselect()</code>	tobit sample selection
<code>extreat()</code>	exogenous discrete treatment
<code>entreat()</code>	endogenous discrete treatment

The main features of the ERM commands are as follows:

1. An ERM command can be used when the regression of interest is embedded in a triangular or recursive model and has right-hand-side endogenous variables for which a reduced-form-type model is specified. Thus, no structural feedback from the endogenous regressors to the left-hand-side variable of interest is specified.
2. The full model is parametric, the joint distribution of the errors in the model is multivariate (bivariate) normal, and estimation is by ML methods that use quadrature integration methods.
3. An ERM command is flexible because it allows one to simultaneously address multiple complications, such as both endogeneity and selection.
4. When the model is linear but not recursive, it must be triangularized before one applies an ERM command. This happens when the specified model involves simultaneous dependence. The remedy is to eliminate some endogenous variables by substituting reduced forms. Triangulation is not feasible for a nonlinear model, such as one involving, say, the probit equation, because in this case there is no closed-form reduced-form model to substitute. For details, see [ERM] **eregress**.

The flexibility of the ERM commands derives from the user's ability to specify and fit a wider range of models by adding or modifying subcommands. For example, the command

```
eregress y1 x, endogenous(y2 = z1 x) select(s1 = z2 x, probit)
```

fits a linear regression model with endogenous regressor y_2 and an endogenous binary selection variable s_1 , where the instruments z_1 for y_2 and z_2 for s_1 are excluded from the outcome equation for y_1 .

Simpler applications of the ERM commands are equivalent to other model-specific commands. Thus, elsewhere in the book, ERM commands have been used selectively as an ML-based alternative to several other commands, including `ivprobit`, `ivtobit`, and `ivregrss`, `liml`.

Table 23.4 presents some commands presented in preceding chapters and the corresponding ERM command. The prefix `e` tells the user that the command is an ERM command. The syntax has the equation specification followed by the subcommand that specifies the reduced form with the instrumental variables z and x . For command syntax, see the relevant help files.

Table 23.4. Selected ERM commands for handling endogeneity and selection

Stata model command	Stata ERM command equivalent
Linear regression with endogenous regressor <code>ivregrss liml y1 x (y2 = z)</code>	<code>eregress y1 x, endogenous(y2 = z x)</code>
Binary probit with endogenous regressor <code>ivprobit y1 x (y2 = z)</code>	<code>eprobit y1 x, endogenous(y2 = z x)</code>
Tobit model with endogenous regressors <code>ivtobit y1 x, (y2 = z), ll(0) ul(#)</code>	<code>eintreg y1xll y1xul x, endogenous(y2 = z x)</code>
Linear regression with sample selection <code>heckman y x, select(s1 = x z)</code>	<code>eregress y x, select(s1 = x z)</code>
Probit model with sample selection <code>heckprobit y x, select(s1 = x z)</code>	<code>eprobit y x, select(s1 = x z)</code>
Ordered probit with sample selection <code>heckoprobit y x, select(s1 = x z)</code>	<code>eoprobit y x, select(s1 = x z)</code>

Postestimation commands for ERM commands include `margins`. If the model is fit with the option `vce(robust)`, then the `vce(unconditional)` option of `margins` gives unconditional standard errors that additionally allow for variation in the regressors; see section 13.7.9.

A detailed application of `eregress`, `entreat()` is given in section [25.3.3](#), where it is used to compute treatment effects for a continuous outcome when there are four ordered levels of treatment and these treatment levels are endogenous.

23.7.2 The `eprobit` command

As an example of the complications handled by ERM commands, we consider the simplest forms of the `eprobit` command and, where appropriate, compare the command with related Stata commands.

The `eprobit` command without an option yields the same results as the `probit` command, so `eprobit y x` is equivalent to `probit y x`.

The `eprobit, endogenous()` command estimates by ML a probit model with regressors that include one or more continuous endogenous regressors. The model is that given in section [17.9.2](#). The `ivprobit` command computes the same ML estimator and additionally provides the option of a two-step estimator but is restricted to only one endogenous regressor.

The `eprobit, select()` command estimates by ML a probit outcome model with the complication of binary selection. The probit outcome is observed only for a subset of the population determined by a second probit model with latent normal errors correlated across the two equations. The `heckprob` command computes the same ML estimator.

The `eprobit, tobitselect()` command is similar to the `eprobit, select()` command, except that selection is determined by a continuous variable restricted to the range (l_i, u_i) , rather than a binary indicator variable.

The `eprobit, extreat()` command applies the treatment-effects methods of chapter [24](#) to a binary outcome. In this framework, there are one or more treatment variables, and we fit a separate probit model for each distinct value of the treatment variables. This is essentially the regression-adjustment model

of section [24.6.1](#) adapted to probit regression. The `estat teffects` postestimation command can be used to calculate the average potential-outcome probabilities at each treatment level and to compute average treatment effects.

The `eprobit, entreat()` command extends the `eprobit, extreat()` command by allowing binary treatment and ordered discrete treatments to be endogenous, in which case, respectively, binary probit or ordered probit models are used with multivariate normal errors correlated across the various associated latent variable models.

The `eprobit` command options can be combined. For example, the command `eprobit y x1 x2, entreat(d = x3 x2) endogenous(y2 = x2 x4)` fits a treatment-effects probit model for the binary outcome `y` that is explained by the endogenous discrete treatment `d`, the endogenous continuous regressor `y2`, and the exogenous regressors `x1` and `x2`. Instruments are `x3` for `d` and `x4` for `y2`.

23.7.3 The `eprobit` command application

We consider a probit equation with a single endogenous continuous regressor. We model whether someone has supplementary health insurance (`ins`) using the dataset from chapter [17.9](#). The natural logarithm of income (`linc`) is treated as being endogenous, with instruments the subject's retirement status (`retire`) and the retirement status of the spouse (`sretire`).

We first read in the data and define global macros for regressors in the probit equation and additional regressors used in the reduced-form equation for `linc`.

```
. * Read in data, define globals, and summarize key variables
. qui use mus217hrs, clear
. generate linc = log(hhincome)
(9 missing values generated)
. global xlist female age age2 educyear married hisp white chronic adl hstatusg
. global ivlist retire sretire
```

The MLE assuming joint normal errors is obtained using the `eprobit, endogenous()` command. We have

```

. * Endogenous probit using eprobit ML estimator (an ERM command)
. eprobit ins $xlist, endogenous(linc = $xlist $ivlist) vce(robust) nolog
Extended probit regression                                         Number of obs = 3,197
                                                               Wald chi2(11) = 382.35
Log pseudolikelihood = -5407.7151                               Prob > chi2 = 0.0000

```

		Robust				
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
ins						
female	-.1394073	.049447	-2.82	0.005	-.2363216	-.0424929
age	.2862296	.1280815	2.23	0.025	.0351945	.5372647
age2	-.0021472	.0009318	-2.30	0.021	-.0039735	-.0003209
educyear	.1136882	.0237908	4.78	0.000	.067059	.1603174
married	.7058322	.2377541	2.97	0.003	.2398428	1.171822
hisp	-.5094516	.1049486	-4.85	0.000	-.7151471	-.3037561
white	.1563459	.1035658	1.51	0.131	-.0466394	.3593312
chronic	.0061938	.0275247	0.23	0.822	-.0477536	.0601412
adl	-.1347665	.0349799	-3.85	0.000	-.2033258	-.0662072
hstatusg	.2341791	.070975	3.30	0.001	.0950707	.3732875
linc	-.5338273	.3852044	-1.39	0.166	-1.288814	.2211594
_cons	-10.00788	4.065762	-2.46	0.014	-17.97662	-2.039132
linc						
female	-.0976056	.0305475	-3.20	0.001	-.1574776	-.0377336
age	.2664422	.073175	3.64	0.000	.1230217	.4098626
age2	-.0019031	.0005309	-3.58	0.000	-.0029437	-.0008625
educyear	.0947747	.0044497	21.30	0.000	.0860534	.103496
married	.7847725	.040539	19.36	0.000	.7053176	.8642274
hisp	-.2364977	.0506833	-4.67	0.000	-.3358351	-.1371603
white	.232593	.0360103	6.46	0.000	.1620141	.303172
chronic	-.0387668	.0103384	-3.75	0.000	-.0590297	-.0185039
adl	-.0741455	.0180171	-4.12	0.000	-.1094584	-.0388326
hstatusg	.174596	.0339594	5.14	0.000	.1080368	.2411552
retire	-.0953103	.0283004	-3.37	0.001	-.150778	-.0398426
sretire	-.0326619	.0308983	-1.06	0.290	-.0932216	.0278977
_cons	-7.679616	2.517254	-3.05	0.002	-12.61334	-2.745889
var(e.linc)	.5152062	.0240909			.4700879	.5646548
corr(e.linc, e.ins)	.5879572	.2355274	2.50	0.013	-.0309703	.880964

The last line shows positive and statistically significant (at 5%) correlation between the structural and reduced-form residuals, so there is an endogeneity problem in this specification.

The estimates show a perhaps surprising negative effect of `linc` on insurance choice after controlling for other variables and the endogeneity of `linc`, though the coefficient is statistically insignificant at level 0.05. From results not listed here, regular probit estimation of the same model leads to `linc` having a positive coefficient of 0.347 and being much more precisely estimated with a standard error of 0.040.

Additional examples of the ERM commands appear in section [25.3](#) on treatment effects, where several flexible versions of the command are used to simultaneously handle endogenous regressors and endogenous treatment choice.

23.8 Additional resources

The various commands presented in this chapter have potentially many applications. For details and many examples, see the Stata reference manuals [FMM] *Finite Mixture Models*, [ME] *Mixed Models*, [SEM] *Structural Equation Models*, and [ERM] *Extended Regression Models*. For detailed coverage of QR, see chapter 15.

For comprehensive coverage of mixed models, see [Rabe-Hesketh and Skrondal \(2022\)](#). For SEMS, a standard reference is [Bollen \(1989\)](#). For GSEMS, see [Rabe-Hesketh, Skrondal, and Pickles \(2004\)](#). The methodology of ERMS is presented in [Roodman \(2011\)](#). The reader is referred to the earlier chapters dealing with `ivregress` and `ivtobit` models for empirical illustrations and encouraged to replicate those results using the ERM commands.

23.9 Exercises

1. Consider the gamma regression data used in example 1 of section [23.3](#). Estimate the gamma regression given the one-component conditional mean specification used in that example. Generate predicted values of expenditure, and using a two-way scatter diagram, compare them with observed values as a rough goodness-of-fit test. Are there any indications of lack of fit? Next, graph the kernel density of the residuals from this regression. Check for visual evidence of multimodality of the graph.
2. Continuing with the expenditure data of the preceding exercise, fit a varying probability FMM2 model in which the latent class probability depends upon `totchr` (rather than `female` and `age` as in the example). Compare the fit of this model with that of the constant class probability FMM2 specification.
3. Example 6 of section [23.3](#) concerns a point-mass (zero-inflated Poisson) model for count data. This model has two parts, the second one being the truncated Poisson model. Stata's `fmm` prefix supports estimation of mixtures of truncated Poisson regression. Using the truncated-at-zero version of the same dataset and the same model specification, estimate the truncated Poisson regression, the FMM2, and FMM3 versions of the same. Select the best-fitting model according to AIC and BIC criteria, and compare the estimates of the AME of `totchr` on `er` visits in the three models.
4. Consider the just-identified IV application of section 7.4.4 that uses data from `mus207mepspresdrugs.dta` for older people in U.S. Medicare. We wish to regress `ldrugexp` on an indicator variable for access to additional employer or union-sponsored health insurance (`hi_empunion`), number of total chronic conditions (`totchr`), and socioeconomic variables `age`, `female`, `blhisp`, and `linc`. Variable `hi_empunion` is endogenous, and we have single instrument `ssiratio`, which is the ratio of an individual's social security income to the individual's income from all sources. Obtain IV estimates using the `ivregress` command. Obtain the equivalent estimates using the `gsem` command. In both cases, obtain heteroskedastic-robust standard

errors. Do the two commands give identical coefficient estimates? Do they give identical standard errors?

5. Reconsider the GSEM empirical application of section [23.6.7](#). Carry out estimation and postestimation operations after changing the `nbreg` option in the `gsem` command to `poisson` and applying the parametric constraint to the `numchron` variable. Also, reestimate the specification without constraints, and then apply the LR test of the constraints.

Chapter 24

Randomized control trials and exogenous treatment effects

24.1 Introduction

This chapter and the next are concerned with identification and consistent estimation of treatment-effect (TE) parameters that measure the causal impact of a change in some controllable economic variable, henceforth referred to as the treatment variable, on some policy or program target variable, henceforth referred to as the outcome. We concentrate on empirical aspects of treatment evaluation, along with explanation of these methods. For more details on treatment evaluation, see [Angrist and Pischke \(2009\)](#), [Imbens and Rubin \(2015\)](#), [Wooldridge \(2010\)](#), chap. 21) or [Cameron and Trivedi \(2005\)](#), chap. 25).

The topic of TES is not new in this book. As a parameter that measures the impact of some manipulable variable, a TE is closely related to the marginal effect of a change in a variable. Marginal effects have been discussed at many places in the book and especially in section 13.7. However, much of that discussion was in the context of observational data and estimation based on either a fully parametric or a semiparametric regression model.

The treatment evaluation literature focuses on the case where the treatment level is discrete valued and, furthermore, is most often binary, leading to comparison of treatment with control. An experimental viewpoint is taken as the starting point, rather than the usual econometrics starting point of regression. The potential-outcomes framework, under which an individual has several potential outcomes that vary with the treatment level, is used. The complication is that only one of these potential outcomes, corresponding to the treatment received, can be observed; the remaining potential outcomes are unobserved counterfactuals.

A treatment is usually an intervention. Such an intervention may arise from entirely exogenous and unforeseen changes in variables, often referred to as “natural experiments”, that may be interpreted as “treatments”. Alternatively, an intervention may be designed and implemented in a specific setting with the objective of estimating a “pure” or uncontaminated TE—a case referred to as a social experiment or a randomized control trial

(RCT). Finally, an intervention may be initiated by an agent whose response we wish to study, as is usually the case for observational data.

RCTS follow the format of drug trials in which inference about the effect of an intervention is estimated by comparing the response of randomly selected and treated subjects with that of the randomly selected control individuals who are drawn from the same population as the treated subjects but who were not assigned to the treated group. In a well-designed RCT, the average difference in the outcome of the treated and control groups is attributed to the causal impact of the intervention.

Randomization is a well-established and widely applied methodology in experimental settings that arise naturally in agriculture, biomedical sciences, psychology, and so forth. When suitable observational data are not available, an RCT is motivated partly to generate data and partly to overcome potential problems of observational data. More recently, such methods have been applied also in economics, especially in development economics; see [Glennerster and Takavarasha \(2013\)](#).

Experimental data can be used to test for the impact of treatment using relatively simple statistical methods such as difference-in-means tests. This chapter first describes experimental design and assumptions and issues that affect the power of tests applied to experimental data; for the latter, see also section 11.7. Well-designed experiments attain high power, so we consider features of experiments that affect power of tests based on data from RCTS.

Even when experimental data are available, regression-based methods can have a useful role in estimating TES. First, the inclusion of regressors may lead to more precise estimation of the TE. Second, actual implementation of an RCT can be complex and result in systematic differences in the individuals in treatment and control samples, in which case regression-adjustment (RA) methods such as inverse-probability weighting may be appropriate.

We initially focus on an RCT in this chapter before studying TES using observational data. With observational data, it is necessary to introduce control variables in addition to the treatment variable. Furthermore, to give a causal interpretation to estimates requires the very strong assumption that

sufficient controls are included in models so that conditional on these controls, any selection into treatment is uncorrelated with the potential outcomes. This nontestable assumption of exogenous treatment goes by several names, including unconfoundedness, ignorability, and selection on observables. This chapter covers methods for both RCTs and for observational data in the special case that selection into treatment is assumed to be exogenous. In chapter 25, we consider additional methods of treatment evaluation, including methods for when it is unreasonable to assume that selection is only on observables, so that the treatment is endogenous.

The methods presented in this chapter are illustrated using a sample of data from a recent social experiment—the Oregon Health Insurance Experiment (OHIE). This RCT was a lottery whose winners were granted the option of applying for enrollment in Medicaid, the state health insurance program for low-income individuals. Interest lies in the impact on various health outcomes.

24.2 Potential outcomes

Treated and untreated outcomes cannot be simultaneously observed for a given individual; this fact is a major obstacle for identification and estimation of a causal parameter. It has been labeled the fundamental problem of causal inference; see [Holland \(1986\)](#). The concept of potential outcomes plays a key role in resolving the difficulty.

Let D be the hypothesized cause and y the outcome. Let D change to D_1 from D_0 and y change to y_1 from y_0 . Then given observation of y_1 , no conclusion can be drawn about the causal impact without a hypothesis about what value y would have assumed in the absence of the change in D . This is referred to as the counterfactual, that is, the hypothetical unobserved value that forms the basis of comparison. This can also be viewed as a case of missing data. To resolve the difficulty, the investigator essentially generates the missing observations using a model. For the treated group, the missing data are realizations that would have been observed in the absence of treatment. For the untreated group, they are the realizations that would have been observed had a treatment been assigned. Given a dataset thus expanded, a comparison of such generated observations with observed data then forms the basis of counterfactual causality inference.

Causal inference involves comparison of a factual with a counterfactual outcome. The so-called Rubin causal model —also called the potential-outcome model—deals with causal parameters based on counterfactuals. In that framework, the term “treatment” is used interchangeably with cause, and it is assumed that all members of the target population are potentially exposed to the treatment.

The triplet (y_{1i}, y_{0i}, D_i) , $i = 1, \dots, N$, forms the basis of treatment evaluation. Initially, we consider the treatment D to be homogeneous and binary valued, so D takes the values 1 and 0, respectively, when treatment is or is not received; y_{1i} measures the response for individual i when receiving treatment; and y_{0i} , when not receiving treatment. That is,

$$y_i = \begin{cases} y_{1i} & \text{if } D_i = 1 \\ y_{0i} & \text{if } D_i = 0 \end{cases}$$

Receiving and not receiving treatment are mutually exclusive states, so $(y_{1i} - y_{0i})$ is not observable for any given individual i because only one of the two measures is available, the unavailable hypothetical measure being the counterfactual.

Hence, inference is instead made about the average effect of the treatment and not the effect on any single individual. The average causal effect of $D_i = 1$, relative to $D_i = 0$, is measured by the average treatment effect (ATE), defined as

$$\text{ATE} = E(y|D = 1) - E(y|D = 0)$$

where expectations are with respect to the probability distribution of the random component over the target population. The ATE can be consistently estimated by the sample counterpart $(\bar{y}_1 - \bar{y}_0)$, under the assumption that each observed outcome includes an additive zero-mean random component that is uncorrelated with the treatment assignment.

The ATE parameter can be estimated from models based on experimental data or observational data. The experimental approach involves a random assignment of treatment followed by a comparison of mean outcomes of the treated and control cases, using $(\bar{y}_1 - \bar{y}_0)$. With observational data, as detailed throughout this chapter, additional assumptions are needed, and other individual-specific variables are added to serve as controls. In the potential-outcomes framework, only some variables are considered causal, while most are regarded as controls for extraneous variation in the outcome. An advantage of this framework is that the counterfactual can clearly indicate what should be compared.

The ATE need not be the only parameter of interest. The ATE on the treated (ATET) is

$$\text{ATET} = E(y_1|D = 1) - E(y_0|D = 1)$$

Other quantities of interest include quantile TEs, introduced in section 15.5 and presented in section [24.11](#).

24.3 Randomized control trials

RCTs are often considered the gold standard against which one compares other econometric/statistical approaches to causal inference. The core appeal of the RCT methodology is based on two considerations—identifiability of the TE and computational simplicity.

The strength of RCT as a tool for causal inference is based on three strong assumptions about treatment assignment; see [Imbens and Rubin \(2015\)](#), chap. 3). First, treatment assignment is individualistic, meaning a given subject's probability of receiving treatment does not depend upon the characteristics, outcomes, or treatment assignments of the other subjects. By assumption, there are no spillover effects that could potentially contaminate the comparison of the treated and control groups. Second, all members of the target population can potentially receive the treatment. Hence, there is positive probability that a randomly selected member of the population can receive treatment. Third, the treatment assignment is independent of the potential outcome. This rules out self-selection under which individuals who stand to gain more by receiving treatment are more likely to self-select into it.

It follows that if the RCT is properly planned and correctly implemented, the problem of selection bias is eliminated. In contrast, the problem of self-selection into treatment plagues analyses based on observational data. Given self-selection bias, the estimate of the TE is confounded with (that is, contaminated by) the self-selection effect, and the TE is then not identified. Under an RCT, the treatment is exogenously and randomly applied to a subset of participants whose responses can be compared with those of other individuals who could potentially have received the treatment but did not. Random assignment implies that treatment assignment does not depend upon the observable characteristics of an individual, and hence ignoring them will not generate biases. (If a blocked or stratified experimental design is used, then assignment within each block is random.) An estimate of the TE is based on such a comparison; the details follow in a later section. The RCT methodology ensures that the TE is not confounded with changes that emanate from sources unrelated to the treatment.

The second appealing feature of an RCT is that the estimation of the TE is computationally simpler than is typically the case with observational data. In the simplest of cases, estimation of the ATE parameter reduces to a comparison of the mean response of the treated and untreated groups. In contrast, the corresponding estimates based on observational data are usually based on (often nonlinear) regression that may invoke several auxiliary assumptions.

This apparent simplicity of estimation in an RCT setting, however, becomes more complicated if the underlying assumptions and requirements of an ideal RCT are not achieved. Then, even in an RCT, alternative estimators may need to be used.

24.3.1 Simple RCT setting of difference in means

Consider a simplified setting for evaluating a treatment applied to a group of N subjects who voluntarily consent to participate in the trial. Of these, a randomly selected subset of size N_1 is assigned to the treatment, while N_0 subjects are assigned to the control group that receives no treatment. At the end of the trial, all $N_1 + N_0$ are tested by comparing the average score of the treated group, denoted \bar{y}_1 , with the average of the untreated control group, denoted \bar{y}_0 , to see whether the TE is statistically significant. The comparison is implemented using a two-sample t test.

The test of the null hypothesis of zero effect is a test of differences in means. Given independent and identically distributed (i.i.d.) data in the treatment group and i.i.d. data in the control group, the test is based on the statistic

$$t = \frac{(\bar{y}_1 - \bar{y}_0) - (\mu_1 - \mu_0)}{s_D} \quad (24.1)$$

where s_D , the standard error of $(\bar{y}_1 - \bar{y}_0)$, is given by

$$(24.2)$$

$$s_D = \sqrt{(s_1^2/N_1 + s_0^2/N_0)}$$

and $s_1^2 = \sum(y_{1i} - \bar{y}_1)^2/(N_1 - 1)$ and $s_0^2 = \sum(y_{0i} - \bar{y}_0)^2/(N_0 - 1)$ are, respectively, the usual estimated standard deviations of y_1 in the treated group and y_0 in the control group. The null hypothesis sets $(\mu_1 - \mu_0) = 0$. Large values of t lead to rejection of the null hypothesis.

The test statistic has an approximate Student's t distribution with ν degrees of freedom. The difference-in-means test is detailed in section 3.5.12. The test can be implemented using the command `ttest y, by(D) unequal`, which uses Satterthwaite's approximation, which sets $\nu = \{(s_1^2/N_1) + (s_0^2/N_0)^2\}/\{(s_1^2/n_1)/(N_1 - 1) + (s_0^2/N_0)^2/(N_0 - n_0)\}$. Alternatively, the test can be implemented as a t test of the coefficient of D following the command `regress y D, vce(robust)`, with $\nu = N_1 + N_0 - 2$. In addition to this different value of ν , the `regress` command leads to a slightly different value of t because it uses a slightly different degrees-of-freedom correction in obtaining s_D , the standard error of $(\bar{y}_1 - \bar{y}_0)$; see section 3.5.12.

Use of the $t(\nu)$ distribution for inference is only an approximation. An exact test of $H_0 : \mu_1 - \mu_0$ based on the t statistic can be performed using a permutation test; see section 11.10. To date, this has been rarely done in econometrics studies. A notable exception is [Young \(2019\)](#), who applies randomization inference to many published RCTs and, where relevant, corrects for clustering.

24.3.2 Optimal sample size and power analysis

Running an RCT can be expensive. The goal of experimental design is to choose treated and control groups large enough to detect whether treatment has an effect, allowing for inherent randomness in any experiment, but not unnecessarily large. The design depends in part on the variance of the outcome variable, a parameter about which accurate information may not be available prior to running the trial. Often, a pilot experiment is run to obtain a preliminary estimate of variance.

An RCT tests the null hypothesis of zero impact of intervention against the alternative of a nonzero impact for a two-tailed test or a positive (or negative) impact if a one-tailed test is used. Such tests of hypotheses in the Neyman–Pearson framework are subject to both type I error of rejecting H_0 when it is true and a type II error of failing to reject H_0 when it is false.

Table 24.1 summarizes the notation used for power analysis. Standard practice is to fix type I error by using a conventional significance level α , often 0.05. The probability of a type II error, denoted β , depends on the mean TE size,

$$\delta = \mu_1 - \mu_0 \quad (24.3)$$

where $\mu_1 = E(y_1)$ is the mean in the treatment group and $\mu_0 = E(y_0)$ is the mean in the control group. To reach useful conclusions about interventions, one designs the RCTs to achieve high power against a meaningful alternative, where power $\pi = 1 - \beta$ equals one minus the probability of a type II error. A conventionally preferred level of power is 80%, so $\beta = 0.20$ and $\pi = 0.80$

Table 24.1. Size and power analysis terminology and notation

Description	Symbol
Total sample size	N
Treated sample size	N_1
Control sample size	N_0
Treatment group (mean, variance)	(μ_1, σ_1^2)
Control group (mean, variance)	(μ_0, σ_0^2)
Treatment-effect size	$\delta = \mu_1 - \mu_0$
Significance level	α
Type 2 error probability	β
Power	$\pi = 1 - \beta$

The RCT implementation requires a decision on the minimum detectable mean difference between the treatment and the control group ([List, Sadoff, and Wagner 2011](#)). This is essentially the minimum value of δ that the RCT will be able to detect, given the desired significance level α and power $(1 - \beta)$.

Just like the underlying t test, the power function varies according to whether the variances σ_1^2 and σ_0^2 are known. The alternative hypothesis leads to a translation of the mean. In the case of known variances, inference is based on the normal distribution because a translated mean of the normal leads again to a normal distribution and the power function involves the normal distribution. In the case of unknown variances, inference is based on the t distribution, and a translated mean of the t leads to a noncentral t distribution. For economics applications, the variances are unknown and need to be estimated, so we use the t statistic given in [\(24.1\)](#)–[\(24.2\)](#).

Let $T(\nu, \lambda)$ denote the cumulative distribution function of a noncentral t distribution, with ν degrees of freedom and noncentrality parameter λ , and let $t_{\nu, \alpha}$ denote the area α in the right tail of the usual central $t(\nu)$ distribution [that is, the $(1 - \alpha)$ th quantile]. Then, for given effect size δ and standard error s_D of the mean difference $(\bar{y}_1 - \bar{y}_0)$ defined in [\(24.2\)](#), the power of a two-sided test of $H_0 : \mu_1 = \mu_0$ of size α is defined by

$$\pi(\delta, s_D, \alpha) = 1 - T_{\nu, \lambda}(t_{\nu, \alpha/2}) + T_{\nu, \lambda}(-t_{\nu, \alpha/2})$$

where ν is Satterthwaite's degrees of freedom and $\lambda = \delta/s_D = |\mu_1 - \mu_0|/s_D$ is the noncentrality parameter. For an upper one-sided test, the power at size α is $1 - T_{\nu, \lambda}(-t_{\nu, \alpha})$. And for a lower one-sided test, the power at size α is $T_{\nu, \lambda}(-t_{\nu, \alpha})$.

Power analysis in experimental design sets size α at a particular value. There are then three types of power analysis. First, compute the power π given specified δ and N_1, N_0, s_1 , and s_0 (and hence s_D). Second, compute the minimum effect size δ given specified power π and N_1, N_0, s_1 , and s_0 (and hence s_D). Third, compute the sample sizes N_1 and N_0 given specified

power π , effect size δ , and the sample standard deviations s_1 and s_0 . Standard practice sets significance level to 0.05 and desired power to 0.8 or 0.9.

24.3.3 Sample size and power calculations in Stata

In Stata, power analysis for a two-means test is implemented using the `power twomeans` command. The command has the following syntaxes, which are similar to those for the `power onemean` command, presented in section 11.8.1. To compute power, use

```
power twomeans m1 m2, n(numlist) [options]
```

To compute minimum effect size, use

```
power twomeans m1, n(numlist) power(numlist) [options]
```

And to compute sample sizes, use

```
power twomeans m1 m2 [ , power(numlist) options]
```

where m_1 and m_2 are the means of control and treatment groups corresponding to, respectively, μ_0 and μ_1 in the notation of this chapter.

The defaults set $\alpha = 0.05$ and $\pi = 0.80$ and use the $t(\nu)$ distribution. The option `knownsds` instead uses the standard normal distribution.

The preceding command is for continuous outcomes. For binary outcomes, one can instead use the `power twoproportions` command. Other commands for binary treatments are `power pairedmeans`, `power pairedproportions`, `power twovariances`, `power twocorrelations`, and `power exponential`.

24.3.4 Some examples

We provide a series of examples that are based on hypothetical scenarios in which the treatment and control groups are equal in size, have the same estimated variance, but have different means. Throughout, the size is set at

the default of 0.05. In most examples, we desire power of 0.80; that is, type II error = 0.20. This means that we want to reject the null hypothesis of 0 TE in 80% of the cases when it is false. Under this specification, the larger the difference in the means, the smaller the required sample to achieve our power objective; that is, for a given sample size, a false null is more easily rejected when the sample means are far apart.

Example 1 (sample size): We fix the control group mean at 21 but vary the treatment mean from 23 through 26; that is, the TE δ increases from 2 (less than 10%) to 5 (slightly less than 25%).

```
. * (1) Required sample size when m1=21; m2=23,24,25,26; sd=6
. power twomeans 21 (23(1)26), sd(6)
```

Performing iteration ...

```
Estimated sample sizes for a two-sample means test
t test assuming sd1 = sd2 = sd
H0: m2 = m1 versus Ha: m2 != m1
```

alpha	power	N	N1	N2	delta	m1	m2	sd
.05	.8	286	143	143	2	21	23	6
.05	.8	128	64	64	3	21	24	6
.05	.8	74	37	37	4	21	25	6
.05	.8	48	24	24	5	21	26	6

The larger the TE, other parameters equal, the smaller the required sample size to meet the power objective. The estimated sample size drops from 286 when $\delta = 2$ and to 48 when $\delta = 5$. This implies that when the TE is small, a relatively larger sample is required to detect it with precision.

Example 2 (sample size): This example considers a TE of 3 units but increases the required power from 0.80 to 0.90.

```
. * (2) Required sample size when m1=21; m2=24; sd=6; power=0.9  
. power twomeans 21 24, sd(6) power(.9)
```

Performing iteration ...

Estimated sample sizes for a two-sample means test

t test assuming sd1 = sd2 = sd

H0: m2 = m1 versus Ha: m2 != m1

Study parameters:

```
alpha = 0.0500  
power = 0.9000  
delta = 3.0000  
m1 = 21.0000  
m2 = 24.0000  
sd = 6.0000
```

Estimated sample sizes:

```
N = 172  
N per group = 86
```

Relative to the example 1 calculation with power set at 0.80 when $\delta = 3$, the estimated sample size is 172 instead of 128. Larger samples yield higher power, with other parameters unchanged.

Example 3 (sample size): In this example, we consider the sample size implications of a one-sided test. We do the same calculation as in example 2, with power 0.90, but using a one-sided alternative hypothesis. We expect that a smaller sample will be required to achieve the same power.

```
. * (3) Required sample size when m1=21; m2=24; sd=6; power=0.9; one-sided  
. power twomeans 21 24, sd(6) onesided
```

Performing iteration ...

Estimated sample sizes for a two-sample means test

t test assuming sd1 = sd2 = sd

H0: m2 = m1 versus Ha: m2 > m1

Study parameters:

```
alpha = 0.0500  
power = 0.8000  
delta = 3.0000  
m1 = 21.0000  
m2 = 24.0000  
sd = 6.0000
```

Estimated sample sizes:

```
N = 102  
N per group = 51
```

The estimated total sample size falls from 172 to 102; a one-sided test is more powerful.

Example 4 (minimum TE): The power equation can be used to calculate the minimum TE required to achieve $\pi = 0.80$ when $N = 200$ and the control group mean is set at 21.

```
. * (4) Required minimum TE size when m1=21; sd=6; N=200
. power twomeans 21, sd(6) power(0.8) n(200)
Performing iteration ...
Estimated experimental-group mean for a two-sample means test
t test assuming sd1 = sd2 = sd
H0: m2 = m1  versus  Ha: m2 != m1; m2 > m1
Study parameters:
alpha =      0.0500
power =      0.8000
N =          200
N per group =     100
m1 =        21.0000
sd =        6.0000
Estimated effect size and experimental-group mean:
delta =      2.3888
m2 =        23.3888
```

The estimated TE is 2.39, or around 11% over the base.

Example 5 (power): In this example, we set the TE at 4 units (about 19% above the base level) and estimate the effect on power when the sample size is 100.

```

. * (5) Power when m1=21; m2=25; sd=6; N=100
. power twomeans 21 25, sd(6) n(100)
Estimated power for a two-sample means test
t test assuming sd1 = sd2 = sd
H0: m2 = m1 versus Ha: m2 != m1

Study parameters:
    alpha =      0.0500
        N =        100
    N per group =       50
        delta =     4.0000
            m1 =   21.0000
            m2 =   25.0000
            sd =    6.0000

Estimated power:
    power =     0.9100

```

The estimated power is now 0.91.

Example 6 (power): All else equal, a large variance in the outcomes reduces the power of the test. In this example, we raise the standard deviation by 50% to 9, from 6 in example 5.

```

. * (6) Power when m1=21; m2=25; N=100; sd=9; N=100
. power twomeans 21 25, sd(9) n(100)
Estimated power for a two-sample means test
t test assuming sd1 = sd2 = sd
H0: m2 = m1 versus Ha: m2 != m1

Study parameters:
    alpha =      0.0500
        N =        100
    N per group =       50
        delta =     4.0000
            m1 =   21.0000
            m2 =   25.0000
            sd =    9.0000

Estimated power:
    power =     0.5950

```

The estimated power drops sharply from 0.91 to 0.60.

Most of the results given above are intuitive. Large sample sizes, one-sided tests, large ATES, and low variability are all conducive to high power at

a given desired test size. Conversely, when the ATE is small, then the type II error can be high if the samples are noisy and participation low.

24.3.5 Stratified randomization and clustering

The simple randomization design considered above randomizes the treatment across subjects. In practice, several alternative designs are widely used. For example, rather than randomly selecting classes in a region to be treatment or control, we may restrict analysis to specific schools and, within each school, randomly select some classes to be treated and some to be controls. Usually, the researcher can choose whether to randomize at the individual level or the group level; see [Banerjee and Duflo \(2011\)](#).

Randomization at the group level may be easier and less costly to implement. At the same time, however, this will generally lead to correlated or clustered observations, reducing precision, and variance calculations of the TE should adjust for clustering. Another drawback of randomization at the group level is that spillovers from treatment to comparison groups are more likely, biasing the estimation of TES. For example, a teacher in a treated class may share resultant knowledge with a teacher in a control class.

With cluster correlation, the power calculations need to be adjusted. We assume nonzero intraclass correlation and zero intercluster correlation. Such a dependence pattern could arise from cluster-specific shocks and cluster-specific unobserved heterogeneity. The size and power calculations must be adjusted. The power twomeans options `k1()` and `k2()` denote the number of clusters, and `m1()` and `m2()` denote the cluster sizes for, respectively, control and experimental groups; `kratio()` equals k_2/k_1 and `mratio()` equals m_2/m_1 . The option `rho()` specifies the intraclass correlation.

Example 7 (clustered sample): We suppose that observations are no longer independent. Instead, they are in clusters of 5 observations with intraclass correlation of 0.2 and independence across clusters. We expect to require larger sample sizes given the reduced information when observations are no longer independent.

```

. * (7) Required sample size when m1=21; m2=24; sd=6; cluster size 5; rho=0.2
. power twomeans 21 24, sd(6) m1(5) m2(5) rho(0.2)
Performing iteration ...
Estimated numbers of clusters for a two-sample means test
Cluster randomized design, z test assuming sd1 = sd2 = sd
H0: m2 = m1 versus Ha: m2 != m1
Study parameters:
    alpha =      0.0500
    power =     0.8000
    delta =     3.0000
    m1 =     21.0000
    m2 =     24.0000
    sd =      6.0000
Cluster design:
    M1 =          5
    M2 =          5
    rho =     0.2000
Estimated numbers of clusters and sample sizes:
    K1 =        23
    K2 =        23
    N1 =       115
    N2 =       115

```

The sample size has increased from 128 in example 1 with independent data to $115 + 115 = 230$.

The community-contributed `clsampsi` command ([Batistatou, Roberts, and Roberts 2014](#)) considers treatment at the cluster level in a random-effects model $y_{ij} = \alpha D_j + \mathbf{x}'_{ij}\beta + u_j + \epsilon_{ij}$ that includes regressors. It performs power calculations using the F distribution rather than the chi-squared distribution. Then, the user needs to provide the intracluster correlation coefficient $\sigma_u^2 / (\sigma_u^2 + \sigma_\epsilon^2)$; an estimate can be obtained using existing data and applying the `loneway` command to ordinary least-squares (OLS) residuals.

24.3.6 Limitations of size and power calculations

While size and power calculations are regarded as essential aspects of RCT design, practical application must overcome the difficulty that it requires typically unknown parameter values as inputs. Specifically, the variance of the outcome for the treated and untreated groups are unknown inputs. So,

too, is intracluster-correlation if clustering is an issue. A partial solution to the problem is to use estimates from previous studies as inputs into size and power calculations or to run a pilot study in which such information is collected; see [Glennerster and Takavarasha \(2013\)](#) for details.

24.3.7 Covariate balance

A randomized control experiment should ensure that covariates are reasonably balanced, meaning that individuals in the treatment and control groups are similar; in the case of a stratified design, this balance is within strata. Controlling for other covariates will not affect the consistency of the TE estimate, but it can reduce its variance. Hence, including valid pretreatment regressors (variables that potentially may impact outcome) in the regression will increase power.

It is important that the covariates be balanced because, otherwise, differences between treatment and controls may simply be due to differences in covariates. In fact, the goal of regression decomposition methods given in section 4.6 is to quantify the relative contributions of observable characteristics and unobservable factors, where here the unobservable factors would be attributed to the TE.

If covariates are unbalanced in an RCT, then stratification, blocking, weighting, trimming, and matching methods can be used to improve balance. However, simultaneous stratification in several dimensions can be difficult and may reduce sample size. Stratifying (or blocking) *ex ante* is more efficient than controlling *ex post* because it ensures an equal proportion of treated and untreated units within each block and therefore minimizes variance.

In later sections of this chapter, we extend the analysis beyond randomized experiments. Then the treatment and control groups can be quite unbalanced, and balancing becomes essential. We present a range of methods that have been developed.

24.4 Regression in an RCT

The preceding section considered the case in which the RCT yields data on (y_{1i}, y_{0i}, D_i) . Estimation of the ATE was based on comparison of the average outcomes of treated and nontreated groups. The basic tool of estimation and inference was the two-sample test of difference in means; see section [24.3.1](#).

Now we consider adding regressors as controls to obtain a more efficient estimate of the ATE. With perfect random assignment, these covariates are independent of the treatment, provided they are measured before the treatment, so there is no change in the estimate of the ATE. But if the covariates partially explain the outcome variable, then the variance of the residual is decreased, leading to more efficient estimation.

24.4.1 Adding covariates

Recall from section [24.3.1](#) that the difference-in-means test can be estimated by OLS regression of the outcome y on an intercept and the treatment indicator D .

The simplest way to add covariates \mathbf{x}_i is to add them as regressors in the OLS regression and fit the model

$$y_i = \alpha + \gamma D_i + \mathbf{x}'_i \boldsymbol{\beta} + u_i, \quad i = 1, \dots, N \quad (24.4)$$

Then a test of no TE is a test of the null hypothesis $H_0 : \gamma = 0$.

An important unstated assumption used above is that \mathbf{x} enters the regression linearly. In practice, one may know little about the correct functional form. Hence, the quest for greater power has to be balanced against the risk of misspecified functional form.

It is important to ensure that the exogeneity assumption $E(u|\mathbf{x}, D) = 0$ is satisfied. One way to ensure this is to restrict the set \mathbf{x} to contain only

pretreatment variables. If posttreatment variables are included, there will be a strong suspicion that they may be correlated with u .

In an RCT, \mathbf{x} and D should be uncorrelated. However, some chance correlation in a finite sample cannot be ruled out. Because of sampling variation, the correlation between \mathbf{x} and D in a given sample may not be exactly zero, but it will be approximately so in large samples. Therefore, the regression that includes \mathbf{x} will yield a very similar estimate of γ as the regression without; both estimates are consistent.

If \mathbf{x} has independent predictive value for y , then inclusion of these regressors will usually lead to a more efficient estimate of γ . To see this improvement in efficiency, consider the simplest case of default OLS standard errors with variance matrix $s^2(\mathbf{Z}'\mathbf{Z})^{-1}$, where \mathbf{z} includes an intercept, D and (possibly) \mathbf{x} , and $s^2 = \sum \hat{u}_i^2 / (N - K_{\mathbf{z}})$. Adding covariates leads to little change in the diagonal entry in $(\mathbf{Z}'\mathbf{Z})^{-1}$ for D because by randomization, D is essentially uncorrelated with \mathbf{x} . If the regressors \mathbf{x} make a significant contribution to explaining the variation in y , then $\sum \hat{u}_i^2$ will be significantly smaller, more than offsetting the increase in the degrees of freedom $(N - K_{\mathbf{z}})$. If the additional regressors \mathbf{x} are irrelevant, however, then efficiency can be lower.

Worse still, if the additional regressors are invalid and correlated with D , then their inclusion will bias the estimate of γ ; this may seem unlikely in a carefully designed and implemented trial. This consideration might explain why one might prefer the simple two-sample t test.

24.4.2 Covariates interacted with treatment status

A richer model than (24.4) is one that fully interacts the regressors with treatment status.

$$y_i = \alpha + \gamma D_i + \mathbf{x}'_i \boldsymbol{\beta} + (D_i \mathbf{x}_i)' \boldsymbol{\delta} + u_i, \quad i = 1, \dots, N \quad (24.5)$$

Interpretation of the effect of treatment status is made more difficult because of the interactions that introduce nonlinearity, discussed in

section [24.4.5](#). The model can be fit with the command `regress y i.D#c.x`, where factor variables are used. The `margins D` command yields the sample average predicted outcome when $D = 1$ and when $D = 0$, and the ATE is the difference in these means. The ATE and its standard error can be directly obtained using the `margins, dydx(D)` command.

The interaction model ([24.5](#)) can be shown to yield the same results as the RA method (see section [24.6.1](#)) that runs two separate OLS regressions of y on x , one for the treated sample and one for the control sample; see [Imbens and Rubin \(2015, 127\)](#).

24.4.3 Covariate balance

When TES are estimated under an RCT, or even more importantly, using observational data, attention is given to covariate balance, a principle that ensures that the treated and control groups are comparable in terms of covariates. Absent such balance, a potential exists for different interpretations and paradoxes, such as Simpson's paradox.

Under the regression approach, there is no explicit attempt to obtain covariate balance. Conditional upon including an appropriate set of covariates in the conditional mean function, it is assumed that we have covariate balance. This expectation may not be valid; some categories of individuals may not be equally present in treated and control groups. Inverse-probability weighting and matching methods, presented in section [24.6](#), seek to adjust for this imbalance.

24.4.4 Simulation-based example of regression for an RCT

In this section, we illustrate the usefulness of RA in the context of a simulation-based RCT example with randomly assigned treatment and a single covariate that is uncorrelated with treatment but is correlated with the outcome. We initially fit the simpler model ([24.4](#)) before moving to the interaction model ([24.5](#)).

The variables (x, D) are generated independently. D is a Bernoulli draw with probability 0.5, so assignment to treatment is random. The covariate x

is a draw from the $N(10, 5^2)$ distribution. The dependent variable is generated as $y = 1 + x + 2D + u$, where u is a standard normal draw. The sample size is set at 100.

```
. * Simulated RCT: Exogenous covariate x; similar treat & control sample size
. qui set obs 100
. set seed 10101
. generate x = rnormal(20,5)          // Exogenous regressor
. set seed 10102
. generate D = rbinomial(1,0.5)      // Treatment assignment
. set seed 10103
. generate u = rnormal(0,1)          // Model error
. generate y = 1 + x + 2*D + u      // Outcome varies with treatment
. summarize x D y u
```

Variable	Obs	Mean	Std. dev.	Min	Max
x	100	20.27752	5.234358	4.146819	34.05265
D	100	.48	.5021167	0	1
y	100	22.33587	5.227955	7.290004	35.98082
u	100	.0983549	.9576001	-2.26003	2.745963

```
. pwcorr x D y u, star(0.05)
```

	x	D	y	u
x	1.0000			
D	-0.1034	1.0000		
y	0.9656*	0.0845	1.0000	
u	-0.0860	-0.0224	0.0927	1.0000

The data summary shows that in spite of the design, there is slight correlation (-0.103) between x and D and slight correlation (-0.086) between x and u , though these correlations are not statistically different from 0 at level 0.05. By construction, there is high correlation between y and x , so adding x as a control will greatly improve precision.

Difference in means

The analysis begins with a difference-in-means test.

```
. * ATE: Use ttest command (no controls)
. ttest y, by(D) unequal
```

Two-sample t test with unequal variances

Group	Obs	Mean	Std. err.	Std. dev.	[95% conf. interval]
0	52	21.91365	.7119351	5.133837	20.48438 23.34291
1	48	22.79328	.7713672	5.344189	21.24149 24.34507
Combined	100	22.33587	.5227955	5.227955	21.29853 23.37321
diff		-.8796343	1.049695		-2.9631 1.203831

```
diff = mean(0) - mean(1)                                t = -0.8380
H0: diff = 0                                         Satterthwaite's degrees of freedom = 96.5876
Ha: diff < 0                                         Ha: diff != 0
Pr(T < t) = 0.2021                                     Pr(|T| > |t|) = 0.4041
                                                       Ha: diff > 0
                                                       Pr(T > t) = 0.7979
```

In the generated sample, 48 were randomly assigned to treatment. The ATE is $22.79 - 21.91 = 0.88$, compared with the true value of 2. The `ttest` command uses the second group, here the treated group with $D = 1$, as the reference group. This leads to the sign reversal and estimated difference of -0.88 .

Virtually identical results are obtained by OLS regression of y on D and an intercept.

```
. * ATE: OLS regress y on D (no controls)
. regress y D, vce(robust)
```

Linear regression	Number of obs	=	100
	F(1, 98)	=	0.70
	Prob > F	=	0.4041
	R-squared	=	0.0071
	Root MSE	=	5.2358

y	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]
D	.8796343	1.049643	0.84	0.404	-1.203348 2.962617
_cons	21.91365	.7122145	30.77	0.000	20.50028 23.32701

As expected, the estimated coefficient of D is again 0.8796343, with standard error 1.049643, which differs from that from the `ttest` command in the seventh significant digit because of slightly different degrees-of-freedom

correction. The 95% confidence interval includes 0. The low R^2 of 0.007 confirms that the test will have low power.

Adding covariates

The two-sample t test given above shows that there is very weak evidence of a statistically significant TE, with $p = 0.40$. The main problem is that the test has low power because the underlying relationship between y and D is noisy with correlation 0.0845.

Adding x , which is highly correlated with y though uncorrelated with D , would greatly improve the fit of the model and the power of the test given the data-generating process (DGP) of this example. We begin with the simpler model ([24.4](#)).

Linear regression						
	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]	
x	.9836596	.0167009	58.90	0.000	.9505129	1.016806
D	1.939621	.1918818	10.11	0.000	1.558788	2.320453
_cons	1.45868	.3822349	3.82	0.000	.7000495	2.217311

This boosts the R^2 to 0.97 and produces a TE estimate of 1.9396 that is statistically significant at level 0.05. It is also not significantly different from the DGP value of 2 at level 0.05 because the 95% confidence interval includes 2.

Adding interaction of treatment status with covariates

We now fit the regression model ([24.5](#)) with regressors interacted with the treatment indicator using factor-variable operators.

```
. * OLS regress y on D and x and interaction of D and x
. regress y i.D##c.x, vce(robust) noheader
```

y	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
1.D	1.643602	.7424357	2.21	0.029	.1698785	3.117325
x	.9764098	.0278316	35.08	0.000	.9211645	1.031655
D#c.x						
1	.014617	.0343537	0.43	0.671	-.0535746	.0828087
_cons	1.609437	.6047906	2.66	0.009	.4089372	2.809937

The `margins` command using the predictive margin method of section 13.6 yields the sample average predictions when $D = 1$ and $D = 0$.

```
. * Average predicted outcomes for D=0 and D=1 from interactive regression
. margins D          // POMs
Predictive margins                                         Number of obs = 100
Model VCE: Robust
Expression: Linear prediction, predict()
```

	Delta-method					
	Margin	std. err.	t	P> t	[95% conf. interval]	
D						
0	21.4086	.1467133	145.92	0.000	21.11738	21.69983
1	23.3486	.1252627	186.40	0.000	23.09996	23.59725

The estimated ATE is the difference $23.3486 - 21.4086 = 1.9400$.

The `margins, dydx(D)` command directly yields the ATE estimate.

```
. * ATE: Using margin, dydx command after interactive regression
. margins, dydx(D) // ATE
Average marginal effects                                         Number of obs = 100
Model VCE: Robust
Expression: Linear prediction, predict()
dy/dx wrt: 1.D
```

	Delta-method					
	dy/dx	std. err.	t	P> t	[95% conf. interval]	
1.D	1.939999	.1929133	10.06	0.000	1.557069	2.322929

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

The estimate of 1.9399 is very close to 1.9396 obtained using the simpler model ([24.4](#)), and the precision is similar with standard error 0.193 compared with 0.192. This is expected because the interaction term did not appear in the DGP of this example. The estimate at level 0.05 is statistically significant and is not significantly different from the DGP value of 2.

The same results are obtained using the RA method of section [24.6.1](#) and are more directly obtained using the `teffects ra` command, detailed in section [24.7](#) and illustrated in section [24.9](#). We have

* ATE: Equivalent results using <code>teffects ra</code> command						
. <code>teffects ra (y x) (D)</code> , <code>ate nolog</code>						
Treatment-effects estimation				Number of obs = 100		
Estimator : regression adjustment						
Outcome model : linear						
Treatment model: none						
y		Robust				
		Coefficient	std. err.	z	P> z	[95% conf. interval]
ATE	D (1 vs 0)	1.939999	.1890464	10.26	0.000	1.569475 2.310523
POmean	D 0	21.4086	.5289239	40.48	0.000	20.37193 22.44527

The same estimate of 1.93999 is obtained, while the standard error differs slightly because of estimation of two separate equations for treated and untreated rather than a single equation for both treated and untreated. The option `atet` would yield the estimated ATET.

24.4.5 Marginal effects and nonlinearity

In section 13.7, we explored the connection between TES and marginal effects. This is a fruitful connection to revisit because Stata's powerful `margins` command provides a convenient way to estimate TES in nonlinear models.

The regression framework is very convenient for estimating TES even if the relevant data are derived from an RCT. However, in most cases, this step

should accommodate complications that result from relaxing distributional and functional form assumptions.

In the simpler linear model (24.4) with $y_i = \alpha + \gamma D_i + \mathbf{x}'_i \boldsymbol{\beta} + u_i$, the ATET, ATE, and marginal effect are constant and equivalent, equaling γ . In nonlinear models, irrespective of the assignment mechanism, this equivalence does not hold. Given nonlinearities in variables or parameters, coefficient estimates are harder to interpret. The marginal TE provides the building block for TE.

A marginal TE measures the effect on the conditional mean of y of a change in treatment variable D , which can be calculated by calculus or finite differences depending on whether D is continuous or discrete. When the marginal TE is a function, as in nonlinear models, the function has to be evaluated using one of three common points of evaluation: 1) at sample values of regressors and then averaged; 2) at the sample mean of the regressors; and 3) at representative values of the regressors. These tasks can be accomplished using Stata's powerful `margins` postestimation command, as shown in the example in the preceding subsection.

Another useful tool for computing TEs in nonlinear models is to use the `predict` postestimation command to generate individual-level predictive means (PMS) and predictive margins. Given estimated regression $\hat{y} = \mathbf{x}' \hat{\boldsymbol{\beta}}$, the conditional mean $E(y|\mathbf{x} = \mathbf{x}^*) = \mathbf{x}^* \hat{\boldsymbol{\beta}}$ is called the PM. It is convenient to create profiles of the PM by evaluating it at specified values of \mathbf{x} , say, \mathbf{x}^* , and then contrasting these across treated and control groups. Such PM calculations are easy and flexible to produce. More details are provided in section 13.6.

24.4.6 Cluster correction

If the RCT design involves randomization across clusters, appropriately defined, then (24.4) becomes

$$y_{ig} = \gamma D_i + \mathbf{x}'_{ig} \boldsymbol{\beta} + \nu_g + u_{ig}, \quad i = 1, \dots, N_g; \quad g = 1, \dots, G$$

where each observation is a member of one of G clusters and the g th cluster contains N_g observations, so $N = \sum_g N_g$.

Here the presence of a cluster-specific random effect ν_g implies that the outcomes are conditionally correlated within each cluster but not between clusters. As noted in section 24.3.6, the cluster-induced correlation will reduce the efficiency of estimation of γ . The clustered power calculations assumed equicorrelation within cluster, consistent with a random-effects model with ν_g and u_g i.i.d.

For estimation, however, we can use the usual cluster-robust correction. The commands `teffects` include a `vce(cluster clustvar)` option, where the unique cluster identifier is defined by the stratum level at which the randomization was done. This approach was adopted in making estimating cluster-robust standard errors in section 6.7. Any other form of stratified randomized treatment can be handled in the same way.

24.4.7 Design-based inference

Throughout this book, we have used sampling-based inference based on random sampling of data such as (y_i, D_i, \mathbf{x}_i) or $(\mathbf{y}_g, \mathbf{D}_g, \mathbf{X}_g)$.

Design-based inference additionally controls for randomness due to treatment assignment. [Abadie et al. \(2020\)](#) consider the independence case and find that the usual sampling-based heteroskedastic-robust standard errors can be somewhat conservative. For clustered experiments, [Abadie et al. \(2022\)](#) find that in some settings the usual sampling-based cluster-robust standard errors can be exceptionally conservative and propose two alternative standard error estimators in the case of a binary treatment. The difference in standard errors arises when treatment assignment varies within each cluster and average treatment effects differ across clusters. In that case, the difference additionally increases with the number of observations per cluster and the fraction of the population clusters that are sampled.

When randomized treatment is assigned solely at the cluster level, [Su and Ding \(2021\)](#) propose design-based inference for a variety of estimators, such as weighted regression estimators based on cluster averages.

In this chapter, we consider only sampling-based inference. For RCTs, it provides inference that is valid, albeit potentially quite conservative in some clustered settings. The design-based approach is not presented in detail here because of its newness at the time of writing this text.

24.4.8 RCT as a “gold standard” of treatment evaluation

The RCT methodology has been considered a benchmark for judging methods based on observational data. However, it also has serious critiques; see, for example, [Deaton \(2010\)](#).

In addition to the limitations mentioned in earlier sections, critics of RCT emphasize that, like several other approaches such as the potential-outcomes model used for experimental and observational data, the approach has a black box character. The analysis focuses on estimating the size of the impact but not on the mechanism or the process through which the impact comes about. Whereas an RCT may confirm that the intervention “worked” (or not), it may be less informative about whether it might also work in other different, larger environments, which is an issue of external validity. Econometric models that incorporate additional information about the mechanism behind the impact if reasonably correctly specified can provide a better basis for prediction and extrapolation.

24.5 Treatment evaluation with exogenous treatment

The presentation so far has focused on treatment evaluation for RCTs. Now we consider extension to a wider range of settings, where treatment assignment is not necessarily purely random or purely random within strata.

The idea is to control for individual characteristics in such a way that it is possible to interpret the difference in average outcomes across treatment and control groups or subgroups as being the causal effect of the treatment, even if data are observational rather than generated by an RCT. This greatly widens the possible application of treatment evaluation methods. It rests on assumptions, some nontestable, that we now detail.

Using the potential-outcomes framework, for each individual, we observe treatment D , covariates \mathbf{x} , and one of the two potential outcomes y_0 (if not treated) and y_1 (if treated).

To identify TES, we assume the following

1. **Conditional independence assumption:** Conditional on \mathbf{x} , the treatment assignment and potential outcomes are independent, written as

$$y_0, y_1 \perp D | \mathbf{x}$$

2. **Overlap or matching assumption:** For each value of \mathbf{x} , there are both treated and nontreated cases, written as

$$0 < \Pr(D = 1 | \mathbf{x}) < 1$$

Assumption 1 generalizes purely random assignment for which $y_0, y_1 \perp D$ and is also referred to as the unconfoundedness assumption, the ignorability assumption, or the selection-on-observables assumption. It is equivalent to the assumption that after control for covariates \mathbf{x} , there are no omitted variables that may lead to correlation between treatment and

potential outcomes. For example, if people self-select into a training program, believing it will increase their earnings, then it is assumed that this self-selection is fully captured by the covariates (observable characteristics) and not by unobservables also correlated with the potential outcomes. Consequently, there is assumed to be no selection bias or endogeneity bias. The contrary case of selection on unobservables is considered in the next chapter.

The unconfoundedness assumption is nontestable, analogous to nontestability of the validity of the instrument in a just-identified model. In any particular application, especially one using observational data, one needs to explain why this is a reasonable assumption.

In some cases, assumption 1 can be relaxed to $y_0 \perp D|\mathbf{x}$, which implies conditional independence of participation and y_0 , or relaxed even further to the conditional mean independence assumption

$$\begin{aligned} E(y_1|D = 1, \mathbf{x}) &= E(y_1|D = 0, \mathbf{x}) = E(y_1|\mathbf{x}) \\ E(y_0|D = 1, \mathbf{x}) &= E(y_0|D = 0, \mathbf{x}) = E(y_0|\mathbf{x}) \end{aligned} \tag{24.6}$$

which implies that the mean potential outcomes are unrelated with the treatment assignment.

Assumption 2 can be interpreted to mean that every unit in the sample has a positive probability of receiving treatment and there are no units that are certain to be treated or to be not treated. Furthermore, this probability is bounded away from 0 and 1, so that the regressors \mathbf{x} do not completely determine assignment to treatment or control. The probability of treatment, $\Pr(D = 1|\mathbf{x})$, is called the propensity score.

To these assumptions, we also add an assumption that there are no general equilibrium effects due to the policy intervention. The stable unit-treatment value assumption is that an individual's potential outcome does not vary with treatments assigned to other individuals.

Define δ as the difference between the outcome in the treated and untreated states for an individual, a quantity that is not observable. Then the ATE is the expected value of δ , and the ATET is the expected value of δ given treatment. We have

$$\begin{aligned}\delta &= y_1 - y_0 \\ \text{ATE} &= E(\delta) \\ \text{ATET} &= E(\delta|D = 1)\end{aligned}$$

The ATE parameter is identified under assumptions 1 and 2. The ATET parameter is identified under conditional mean independence (a weaker version of assumption 1) and assumption 2.

Assumptions 1 and 2 hold in a well-designed and executed RCT. They might additionally hold in nonexperimental settings that rely on observational data. Hence, the methods considered in the following sections will also apply to observational data when a convincing case can be made that selection on unobservables is not a major hurdle.

For continuous outcome, the OLS estimator is consistent under the assumptions listed here, provided the covariates \mathbf{x} enter in a sufficiently flexible way. The regression approach is not particularly robust, however, especially if there is substantial difference in average characteristics of treated and controlled individuals, as is often the case for studies using observational data, because it requires correct specification of $E(y|D, \mathbf{x})$. We now present several methods that promise more robust estimates of TES.

24.6 Treatment evaluation methods and estimators

We present the leading methods for estimating TES, where TES are heterogeneous, meaning that different individuals may have a different response to treatment, so the goal is to estimate ATES.

These methods fall into the broad classes of regression, inverse-probability weighting, doubly robust, and matching. We also consider the complications of regressor balance, propensity-score overlap, and blocking. The Stata `teffects` commands that implement these methods are presented in section [24.7](#) and are implemented in section [24.9](#).

We present formulas for binary treatment; see [TE] *Stata Treatment-Effects Reference Manual* for more general formulas that cover multiple treatments. For models of potential outcomes, we focus on predictions using linear regression. More generally, other models might be used. Then $\mathbf{x}'_i \hat{\boldsymbol{\beta}}_1$ and $\mathbf{x}'_i \hat{\boldsymbol{\beta}}_0$ are replaced by predictions \hat{y}_{1i} and \hat{y}_{0i} for continuous outcomes, predicted conditional means $\hat{E}(y_{1i}|\mathbf{x}_i)$ and $\hat{E}(y_{0i}|\mathbf{x}_i)$ for counts, and predicted probabilities for binary and ordered discrete outcomes.

24.6.1 Regression adjustment

A richer model than [\(24.3\)](#), called RA, runs two separate OLS regressions of y on x , one for the treated sample and one for the control sample.

$$\begin{aligned} E(y_{1i}) &= \mathbf{x}'_i \boldsymbol{\beta}_1 && \text{if } D_i = 1 \\ E(y_{0i}) &= \mathbf{x}'_i \boldsymbol{\beta}_0 && \text{if } D_i = 0 \end{aligned}$$

These models are then used to predict outcomes for all observations, not just for the subsample used in estimation. For example, the first regression is used not only to predict y_{1i} for treated observations, in which case y_{1i} is observed, but also to predict y_{1i} for control observations, in which case y_{1i} is not observed. So we are imputing the unobserved counterfactual.

Denote these predictions as, respectively, \hat{y}_{1i} and \hat{y}_{0i} . The averages of these estimates are estimates of the potential-outcome means (POMs). The estimated ATE is the average of the difference between the two estimated POMs. And the estimated ATET is the average of the difference between the two estimated POMs when we average only over the N_1 observations that were treated.

$$\begin{aligned}\widehat{\text{POM}}_1 &= \frac{1}{N} \sum_{i=1}^N \hat{y}_{1i} \\ \widehat{\text{POM}}_0 &= \frac{1}{N} \sum_{i=1}^N \hat{y}_{0i} \\ \widehat{\text{ATE}} &= \frac{1}{N} \sum_{i=1}^N \hat{y}_{1i} - \frac{1}{N} \sum_{i=1}^N \hat{y}_{0i} \\ \widehat{\text{ATET}} &= \frac{1}{N_1} \sum_{i=1:D_i=1}^N \hat{y}_{1i} - \frac{1}{N_1} \sum_{i=1:D_i=1}^N \hat{y}_{0i}\end{aligned}$$

The `teffects ra` command provides these regression-adjusted estimates. The default reports just the ATE estimates; option `pomeans` reports the estimated POMs, and the `atet` option reports the estimated ATET. The same estimates can be obtained using the `eregress` command and the `estat teffects` postestimation command; see section [24.7](#).

For simplicity, we consider only OLS regression. More generally, the RA method used may vary with the outcome; logit regression may be used for a binary outcome, Poisson regression for a count outcome, and so on. Semiparametric and parametric methods could also be used to estimate $E(y_{1i}|\mathbf{x}_i)$ and $E(y_{0i}|\mathbf{x}_i)$.

Regression methods were presented at length in section [24.4](#), in the context of an RCT with random assignment of treatment. Then the addition of regressors is relatively innocuous and is actually unnecessary; the motivation is to obtain a more precise estimate of the TE. When RA is used with observational data, however, the choice of covariates and functional form becomes much more important because covariates are being used to ensure that treatment assignment is random after including the covariates as controls. Thus, the regression estimator is not a robust estimator in most observational data settings. The following estimators may be better.

24.6.2 Inverse-probability weighting

The probability of receiving treatment may vary across individuals. This is very likely when observational data are used and can even happen in an RCT such as that analyzed in sections [24.8](#) and [24.9](#).

Inverse-probability weighted (IPW) regression handles this complication by reweighting the data prior to regression, so that weighted observations more nearly satisfy the assumption of equal probability of receiving treatment. The weight assigned to a subject is the reciprocal of the probability of receiving treatment. The method was used to handle missing data due to panel attrition in section [19.10.3](#).

Denote for individual i the probability of treatment, the propensity score, by

$$p(\mathbf{z}_i) = \Pr(D_i = 1 | \mathbf{z}_i)$$

and let $1 - p(\mathbf{z}_i)$ denote the probability of no treatment. Note that we use \mathbf{z} for the propensity-score variables to distinguish them from the controls \mathbf{x} used in RA methods. Some or all variables may be in both \mathbf{x} and \mathbf{z} .

The IPW estimate of the ATE is given by

$$\widehat{\text{ATE}} = \frac{1}{N} \sum_{i=1}^N \frac{D_i}{p(\mathbf{z}_i)} \times y_i - \frac{1}{N} \sum_{i=1}^N \frac{1 - D_i}{1 - p(\mathbf{z}_i)} \times y_i$$

For an RCT with constant treatment assignment probabilities $p(z_i) = p$ and with treated and control samples of sizes $N_1 = pN$ and $N_0 = (1 - p)N$, the formula simplifies to the difference in means $\bar{y}_1 - \bar{y}_0$. The `teffects ipw` command implements this estimator.

A brief justification for this method is the following. Consider the first term, and note that $D_i y_i = D_i y_{1i}$ because $D_i y_{0i} = 0$. Conditioning on the

controls \mathbf{z} , we have

$$E\left(\frac{Dy}{p(\mathbf{z})}|\mathbf{z}\right) = E\left(\frac{Dy_1}{p(\mathbf{z})}|\mathbf{z}\right) = E\left(\frac{D}{p(\mathbf{z})}|\mathbf{z}\right) \times E(y_1|\mathbf{z}) = E(y_1|\mathbf{z})$$

where the second-to-last equality requires the conditional mean assumption (24.6) and the last equality uses $E(D|\mathbf{z}) = p(\mathbf{z})$ for the binary variable \mathbf{z} . By similar algebra, $E[(1 - D)y/\{1 - p(\mathbf{z})\}|\mathbf{z}] = E(y_0|\mathbf{z})$.

Implementation uses a flexible model for the propensity score $p(\mathbf{z})$. Logit or probit models are often used, though semiparametric and nonparametric models can also be used.

The model that generates $\hat{p}(\mathbf{z})$ needs to provide a good fit to the data; interpretation and testing of coefficients is not the objective. To realize a good fit, one commonly uses a flexible specification that includes powers of \mathbf{z} and interactions. This implies that the conditional probability model is typically not parsimonious and there is a risk of overfitting. For example, in the extreme case of more regressors than observations, we will necessarily perfectly fit the data. Methods for choosing a subset regression from the full set of potential regressors include stepwise subset selection and the lasso; see section 28.3. The key objective is to drop variables that contribute little to the fit of the model and hence can be dropped at small risk of misspecification.

24.6.3 Propensity-score overlap

Limitations of IPW derive from a potentially poorly specified conditional probability model and from numerical instability that may arise from having many fitted values in a small neighborhood of 0 or 1 because these values are given larger weight.

Instability may arise also from the failure of the overlap assumption; that is, $0 < p(\mathbf{z}_i) < 1$. Such failure can happen even in a correctly specified conditional probability model. The `teoverlap` command provides checks and a test of the overlap assumption after weighting.

Similarly, if propensity-score matching is used (see section [24.6.7](#)), we need the distributions of the propensity scores by treatment status to be similar.

24.6.4 Regressor balance

In the simplest RCT, treatment assignment is independent of any covariates, so covariates are independent of the outcome. In observational data, by contrast, treatment assignment will be related to covariates that also affect the outcome of interest. A well-specified model for IPW or matching should balance the covariates.

For a single regressor z , let \bar{z}_1 and \bar{z}_0 denote the treated and control sample means, and let \bar{s}_{z1}^2 and \bar{s}_{z0}^2 denote the corresponding sample variances. Two measures of regressor difference across treatment groups are the standardized difference, $(\bar{z}_1 - \bar{z}_0)/\sqrt{(\bar{s}_{z1}^2 + \bar{s}_{z0}^2)/2}$, and the variance ratio $\bar{s}_{z1}^2/\bar{s}_{z0}^2$.

For IPW estimators, we want to find that, after inverse-probability weighting, the standardized difference of each regressor is close to 0 and that the variance ratio is close to 1. Let p_i denote the propensity score $p(\mathbf{z}_i)$ and z_i denote a single regressor that is an element of \mathbf{z}_i . Then the `tebalance` command gives the weighted mean and variance for the treated and controls defined as

$$\begin{aligned}\bar{z}_{1,w} &= \frac{1}{N_1} \sum_{i=1}^N D_i w_i z_i & \bar{z}_{0,w} &= \frac{1}{N_0} \sum_{i=1}^N (1 - D_i) w_i z_i \\ s_{1,w}^2 &= \frac{1}{N_1 - 1} \sum_{i=1}^N D_i w_i (z_i - \bar{z}_{1,w})^2 & s_{0,w}^2 &= \frac{1}{N_0 - 1} \sum_{i=1}^N (1 - D_i) w_i (z_i - \bar{z}_{0,w})^2 \\ w_i &= 1/p_i & w_i &= 1/(1 - p_i)\end{aligned}$$

For nearest-neighbor matching estimators, presented later, the corresponding formulas for $\bar{z}_{1,w}$, $\bar{z}_{0,w}$, $s_{1,w}^2$, and $s_{0,w}^2$ are given by [TE] `tebalance`.

24.6.5 Doubly robust augmented IPW and IPW regression adjustment

Two different hybrid methods combine regression models for the potential outcomes with inverse-probability weighting. These two methods have the advantage of being doubly robust because consistency requires either that the models for the potential outcomes be correctly specified or that the propensity-score model be correctly specified. So one of these can be misspecified. By contrast, the RA estimator uses only potential-outcome models, and these must be correctly specified, and the IPW estimator uses only a propensity-score model, and this must be correctly specified.

The augmented inverse-probability weighted (AIPW) estimator modifies the RA estimator by adding a weighted residual term and, for the linear model, estimates the potential outcomes by

$$\begin{aligned}\widetilde{\text{POM}}_1 &= \frac{1}{N} \sum_{i=1}^N \frac{D_i (y_i - \mathbf{x}'_i \hat{\beta}_1)}{\widehat{p}(\mathbf{z}_i)} - \mathbf{x}'_i \hat{\beta}_1 \\ \widetilde{\text{POM}}_0 &= \frac{1}{N} \sum_{i=1}^N \frac{(1 - D_i) (y_i - \mathbf{x}'_i \hat{\beta}_0)}{1 - \widehat{p}(\mathbf{z}_i)} - \mathbf{x}'_i \hat{\beta}_0\end{aligned}$$

where $\widehat{p}(\mathbf{z}_i)$ is a first-step estimate of $\Pr(D_i = 1 | \mathbf{x}_i)$ such as from a logit model and $\hat{\beta}_1$ and $\hat{\beta}_0$ are the RA estimates given in section [24.6.1](#). The AIPW estimate of the ATE equals $\widetilde{\text{POM}}_1 - \widetilde{\text{POM}}_0$. The `teffects aipw` command implements this estimator.

The IPW-RA estimator, presented in [Wooldridge \(2010, 930\)](#), modifies the RA estimator by obtaining predictions from regressions that weight observations by the inverse probabilities. For linear models, we obtain estimates β_1 and β_0 by the separate weighted least-squares estimations that minimize

$$\begin{aligned}\sum_{i=1}^N w_i (y_{1i} - \mathbf{x}'_i \beta_1)^2 \quad w_i &= 1/\widehat{p}(\mathbf{z}_i) && \text{if } D_i = 1 \\ \sum_{i=1}^N w_i (y_{0i} - \mathbf{x}'_i \beta_0)^2 \quad w_i &= 1 / \{1 - \widehat{p}(\mathbf{z}_i)\} && \text{if } D_i = 0\end{aligned}$$

where $\hat{p}(\mathbf{z}_i)$ is a first-step estimate of $\Pr(D_i = 1 | \mathbf{z}_i)$ such as from a logit model. The IPW-RA estimate of the ATE then equals

$(1/N) \sum_{i=1}^N \mathbf{x}'_i \hat{\beta}_1 - (1/N) \sum_{i=1}^N \mathbf{x}'_i \hat{\beta}_0$, where the two sums are over the entire sample. The `teffects ipwra` command implements this estimator.

While both the AIPW and IPW-RA estimators are doubly robust, the term “doubly robust estimate of ATE” usually refers to the AIPW estimator.

The AIPW estimator satisfies an orthogonality condition that enables use of the lasso to select a subset of potential control variables; see section 24.6.10.

24.6.6 Matching methods

The essential idea underlying matching methods is that one wants to construct a cell whose occupants constitute a matched group based on control variables. Given a matched set, the average cell difference in treated and untreated outcomes can be computed. An exact method would be to have one-to-one matching, in which every treated subject has an untreated counterpart with identical control variables. To implement an exact match, one must use discrete regressors. In practice, one also has continuous regressors for which exact matching is not feasible. Hence, exact matching is too stringent a condition when there are many control variables or even just a few control variables if they take many different values.

In general, the following issues have to be addressed: regressors include both discrete and continuous variables; some cells may be sparse or even empty; and matching may be one to one or one to many. The larger the number of regressors in the model, the more compelling these issues become.

A successful match means that there is at least one untreated subject who matches a treated subject, that is, a counterfactual exists. Whether there is a match is determined by applying a matching criterion that is a measure of the distance between the treated and untreated subjects. Given many control variables, we face a dimensionality problem.

One solution to the dimensionality problem is to replace the regressors by a one-dimensional function of the regressors and use the value of the function to define a match. The propensity score, considered in the next section, is one such matching criterion.

Alternatively, one can use a combination of exact matching for some discrete regressors and closeness in distance, often Euclidean distance, for other regressors. This nearest-neighbor matching (NNM) estimator is presented in section [24.6.9](#).

For any given subject, a specified matching criterion may not be satisfied. This means that there is no counterfactual available for estimating the TE. This could happen if there is insufficient overlap of the distribution of regressor values between the treatment groups. In the interests of obtaining better balance, or sufficient overlap, such an observation may be dropped. The resulting trimmed (smaller) sample is expected to yield a less biased estimate of the TE. Smaller bias is traded off against a wider confidence interval resulting from higher variance due to shrinkage in sample size.

Before presenting matching methods, we note that in preceding sections, there was reason to distinguish between control variables x used in RA and control variables z used in inverse-probability weighting because x and z need not coincide. For matching methods, any control variables are used solely for matching, and we use x to denote the variables used in matching.

24.6.7 Propensity-score matching

An inexact matching method is based on the propensity score estimated for each subject in the sample. The appeal of propensity-score matching (PSM) derives from the established result of [Rosenbaum and Rubin \(1983\)](#), who showed that if treated and untreated units have the same propensity score, then this is equivalent to them having the same distribution of the regressors in the conditional probability model that generates the propensity scores.

The propensity score, the conditional probability of receiving treatment, is presented in sections [24.6.2](#) and [24.6.3](#). As already noted, here we use x to denote the matching variables, so the propensity score is $p(x_i) = \Pr(D_i = 1|x_i)$ and is estimated by binary outcome regression of D

on \mathbf{x} using the full sample. The fitted values $\hat{p}(\mathbf{x}_i)$ are then partitioned into subsets that define matched groups.

Stata implements PSM-based treatment evaluation using the `teffects psmatch` command. The specific procedure follows a two-step method of [Abadie and Imbens \(2006\)](#). All treated and control units are matched with replacement, so different treated units may share a common matched untreated unit. The match may be one to one or one to many. The Euclidean metric is used to find the closest match or matches. The N -sized matched sample can then be used to estimate the ATE and ATET. Because PSM is based on an estimated measure, the variance estimate of the TE needs to be adjusted for this additional source of variation; see [TE] *Stata Treatment-Effects Reference Manual*.

24.6.8 Blocking and stratification

Given the propensity scores, the next step is to generate matches such that a balanced sample with satisfactory overlap is obtained. This requires some form of bracketing (or smoothing) to create matched pairs or matched sets. Stratification or interval matching is based on the idea of dividing the range of variation of the propensity score in intervals such that within each interval, the treated and control units have, on the average, the same propensity score. The ATE is the weighted average of these differences.

Denote by b the blocks defined over intervals of propensity score. Then the TE within b th block is defined as

$$\text{ATE}_b = \frac{1}{N_{1b}} \sum_{i \in I(b)} D_i y_i - \frac{1}{N_{0b}} \sum_{i \in I(b)} (1 - D_i) y_i$$

where $I(b)$ is the set of units in block b , N_{1b} is the number of treated units in the b th block, and N_{0b} is the number of control units in the b th block. Then the TE based on stratification is defined as $\text{ATE}^S = \sum_{b=1}^B \text{ATE}_b \times \{\sum_{i \in I(b)} D_i / (\sum_{\forall i} D_i)\}$, where the weight for each block is given by the corresponding fraction of treated units and where B is the total number

of blocks. The overall TE is a weighted average of block-specific TES. The same approach is used to estimate ATET, but the averaging is over the subset of treated individuals only.

An alternative estimation method is regression based. Given the blocked data structure, a linear regression of y_i on an intercept, D_i and \mathbf{x}_i , is estimated for $i \in I(b)$, and the block-specific ATE_b is given by the coefficient of D . The overall TE, ATE, is the weighted sum $\sum_{b=1}^B w_b \text{ATE}_b$, where the weights w_b are either the proportion of units in the block or the proportion of treated units in the block.

The number of blocks B and the boundary points of each block, that is, the interval width, are choice parameters. To generate a matching set, one may specify a minimum number m of required matches. For every observation $i = 1, \dots, N$, such that for every i receiving treatment 1 or 0, there are $h_i \geq m$ matching observations in the block. The matched set consists of the propensity score $\hat{p}(\mathbf{x}_i)$ and h_i propensity scores of subjects who received the other treatment.

Specifying a small value of m means that the matched set is closer to the i th observation. This implies a smaller bias in the estimate of ATE but a larger variance. Increasing m implies reducing the variance of the estimate potentially at the cost of increasing the bias.

24.6.9 Nearest neighbors matching

A second alternative to exact matching is NNM. This can also be interpreted as a variant of PSM. The NNM is obtained by creating a matched set based on the closeness of the vector of regressors \mathbf{x}_i to the vector \mathbf{x}_j , where the subject j received the other treatment. To make this operational, we need to choose a metric of closeness of two vectors.

A standard choice is the scaled or weighted Euclidean (“Mahalanobis”) distance metric, defined as $(\mathbf{x}_i - \mathbf{x}_j)' \mathbf{S}^{-1} (\mathbf{x}_i - \mathbf{x}_j)$, where $(\mathbf{x}_i - \mathbf{x}_j)$ is the difference in the vector of K regressors and \mathbf{S} is the $K \times K$ matrix of variances and covariances of elements of \mathbf{x} . As in the case of PSM, a minimum number of NNMs may be specified.

A matching criterion is based on a measure of closeness of two observations. A more stringent criterion defines a smaller catchment area for an acceptable match. In the terminology of nonparametric statistics, a criterion is characterized by a bandwidth parameter; see section 2.6.6. The choice of bandwidth, termed a “caliper” in matching applications, is the most important component in selecting the acceptable matching observations. Choosing m nearest-neighbor observations as matches also implicitly uses a bandwidth.

Another variant of NNM is radius matching in which $A_i\{\hat{p}(\mathbf{x})\} = \{\hat{p}_j : (\hat{p}_i - \hat{p}_j)^2 < r\}$ defines the neighborhood based on estimated propensity scores. This means that all control cases with estimated propensity scores falling within radius r are matched to the i th treated case.

The counterfactual is generated by taking a weighted average of the outcomes in the matched set of subjects who received the other treatment. The TE for subject i who received treatment 1 is $\hat{\delta}_{i,\text{NNM}} = (y_i - \bar{y}_{A_i})$, where \bar{y}_{A_i} is the weighted average of outcomes in the reference group consisting of the nearest-neighboring subjects who received the treatment 0. As in the case of PSM, both ATE and ATET can be computed using the estimates $\hat{\delta}_{i,\text{NNM}}$.

This estimator of ATE (and ATET) is potentially biased if matching is based on more than one continuous variable. A second step makes a regression-based bias correction to the estimated potential outcome. The bias-corrected ATE is estimated using bias-adjusted estimates of the potential outcome.

The `teffects nnmatch` command implements PSM-based treatment evaluation.

24.6.10 Machine-learning methods

The methods of this chapter rely on the assumption of unconfoundedness. These methods become more plausible the better the set of control variables used.

Section 28.8 presents in detail the use of machine-learning methods to accomplish this. These methods overcome the overfitting inherent in data

mining by basing estimation on moment conditions with an orthogonalization property (see section [28.8.8](#)) and using sample splitting; see section [28.8.9](#).

Section [28.8.3](#) presents machine-learning methods for estimating α in the simple linear model [\(24.4\)](#); in that case, the TES are homogeneous.

Section [28.9.4](#) presents machine-learning methods for estimating ATE in the heterogeneous effects model using the doubly robust AIPW estimator. Many more applications will be developed over time.

24.6.11 Assessing unconfoundedness

TES analysis with observational data relies on the assumption of unconfoundedness. This is a nontestable assumption, meaning that one cannot confirm that this assumption is valid. [Imbens and Rubin \(2015](#), chap. 21) present several tests whose failure rejects the plausibility of unconfoundedness, even though their passing does not confirm unconfoundedness. The ability to perform such checks is very dependent on the problem at hand.

Some tests rely on changing the analysis in such a way that the estimated TE should be zero. Such changes should not be pure noise, so $y_0, y_1 \perp D$, but should instead be ones that require controlling for the covariates used in the original analysis.

One method introduces pseudo-outcome variables that have a nonzero ATE without controls but should have zero ATE after controlling for observables using the same variables and methods as used in the original analysis. For example, if we had data on pretreatment levels of y for the treated and control individuals, then we could repeat the analysis using one or more of these pretreatment levels as the outcome variable. Because these outcomes occurred before the time of the treatment, there should be no TE. (And if in this case y_{-1} , say, is already included in the control variables, then we could use y_{-2} as the pseudo-outcome.)

A second method requires being able to partition the control individuals into two groups, one of which has characteristics reasonably similar to the

treated group. Then we expect a zero ATE when we treat one of the two control groups as the treated group.

Other tests are ones of the robustness of results to changes in control variables that should lead to little change in the estimated ATE. For example, if data on y are available for several pretreatment periods and are used as control variates, then adding or dropping one of these pretreatment values should make little difference.

24.6.12 Robust confidence intervals for ATE

Many of the TES estimates are obtained by joint estimation of several equations, where each equation involves an m -estimator. The `teffects` commands obtain standard errors by stacking the equations and applying standard asymptotic results to the system.

For example, consider estimating the ATE for a continuous outcome with linear RA model for the outcome variable and a logit model for a binary treatment. Suppose this model is fit using the command `teffects ipwra (y $x) (D $z, logit)`, where `$x` and `$z` are global macros defining relevant variable lists.

The model parameters are estimated using the following equations:

$$\begin{aligned}
 \text{ATE} & \quad \frac{1}{N} \sum_{i=1}^N \left(\mathbf{x}'_i \hat{\boldsymbol{\beta}}_1 - \mathbf{x}'_i \hat{\boldsymbol{\beta}}_0 \right) - \hat{\alpha} = 0 \\
 \text{Potential outcome } (D_i = 1) & \quad \frac{1}{N} \sum_{i=1}^N \frac{D_i \mathbf{x}_i \left(y_i - \mathbf{x}'_i \hat{\boldsymbol{\beta}}_1 \right)}{(N_1/N) \times \Lambda(\mathbf{z}_i, \hat{\gamma})} = \mathbf{0} \\
 \text{Potential outcome } (D_i = 0) & \quad \frac{1}{N} \sum_{i=1}^N \frac{(1 - D_i) \mathbf{x}_i \left(y_i - \mathbf{x}'_i \hat{\boldsymbol{\beta}}_0 \right)}{(N_0/N) \times \{1 - \Lambda(\mathbf{z}_i, \hat{\gamma})\}} = \mathbf{0} \\
 \text{Treatment probability} & \quad \frac{1}{N} \sum_{i=1}^N \{D_i - \Lambda(\mathbf{z}_i, \hat{\gamma})\} \mathbf{z}_i = \mathbf{0}
 \end{aligned}$$

The bottom equation gives the first-order conditions for logit regression of treatment D_i on regressors \mathbf{z}_i . The outcome $D_i = 1$ equation gives the first-order conditions following weighted least-squares regression of the outcome y_i on regressors \mathbf{x}_i , where the weights are the inverse of the predicted probability (propensity score) from the logit model that $D_i = 1$, with additional weight term N_1/N . The outcome $D_i = 0$ equation is a similar equation for untreated observations. The top equation is simply a rewriting of $\widehat{\text{ATE}} = (1/N) \sum_i \mathbf{x}'_i \widehat{\boldsymbol{\beta}}_1 - (1/N) \sum_i \mathbf{x}'_i \widehat{\boldsymbol{\beta}}_0$, and $\widehat{\alpha}$ is the ATE estimate.

Stacking all four equations yields the system of first-order conditions

$$\frac{1}{N} \sum_{i=1}^N \mathbf{s} \left(y_i, D_i, \mathbf{x}_i, \mathbf{z}_i, \widehat{\alpha}, \widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\beta}}_0, \widehat{\gamma} \right) = \mathbf{0}$$

or, more simply, $(1/N) \sum_{i=1}^N \mathbf{s}_i(\widehat{\boldsymbol{\theta}}) = \mathbf{0}$ where $\widehat{\boldsymbol{\theta}} = (\widehat{\alpha}, \widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\beta}}_0, \widehat{\gamma})$.

This system is just identified, so the estimator is an estimating equations estimator, or, equivalently, a just-identified generalized method of moments estimator. Assuming that $E\{\mathbf{s}(y_i, D_i, \mathbf{x}_i, \mathbf{z}_i, \boldsymbol{\theta})\} = \mathbf{0}$, the estimator $\widehat{\boldsymbol{\theta}}$ is consistent for $\boldsymbol{\theta}$ and is asymptotically normally distributed. For independent observations, the heteroskedastic–robust variance–covariance matrix estimate is

$$\widehat{V}_{\text{rob}}(\widehat{\boldsymbol{\theta}}) = \left(\sum_{i=1}^N \frac{\partial \mathbf{s}_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}'} \Big|_{\widehat{\boldsymbol{\theta}}} \right)^{-1} \sum_{i=1}^N \mathbf{s}_i(\widehat{\boldsymbol{\theta}}) \mathbf{s}_i(\widehat{\boldsymbol{\theta}})' \left(\sum_{i=1}^N \frac{\partial \mathbf{s}_i'(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \Big|_{\widehat{\boldsymbol{\theta}}} \right)^{-1}$$

A cluster–robust version replaces the middle matrix with $\sum_{i=1}^N \sum_{j=1}^N \delta_{ij} \mathbf{s}_i(\widehat{\boldsymbol{\theta}}) \mathbf{s}_j(\widehat{\boldsymbol{\theta}})'$, where δ_{ij} is an indicator variable equal to 1 if observations i and j are in the same cluster.

Note that while we need $E\{\mathbf{s}(y, D, \mathbf{x}, \mathbf{z}, \boldsymbol{\theta})\} = \mathbf{0}$ for all parameters to be consistently estimated, the IPW-RA estimator of ATE is consistent even if one

of the RA equation or the treatment probability equation is misspecified. Variations of this example include changing the first equation appropriately to obtain estimates of the ATET and POM. For a multivalued treatment, say, with m treatments, the first equation will become $(m - 1)$ equations (in the case of ATE and ATET), the second and third equations will become m equations, and the treatment probability model may be a multinomial logit (MNL) model. This is the method used by the `teffects` commands; see *Methods and Formulas of [TE] teffects aipw* for a very general treatment.

For models not covered by the `teffects` command, for example, using a complementary log–log model for the treatment probability, one can estimate each model sequentially and obtain standard errors by a two-step bootstrap, as in section 12.4. An alternative method to obtain the standard errors is to again use a “stacked” moment-based approach and use the `gmm` command; section 13.3.11 provides an example.

The preceding analysis assumes that estimators are standard root- N consistent and asymptotically normal. For matching estimators, the asymptotics are nonstandard, and alternative methods are used to obtain the standard errors.

24.7 Stata commands for treatment evaluation

Stata has several commands for treatment evaluation. The starting point for models that satisfy unconfoundedness is the `teffects` commands. We additionally briefly summarize other commands for treatment evaluation.

24.7.1 The `teffects` commands and `telasso` command

Table 24.2 displays the standard commands for basic TE analysis for treatment when the treatment is exogenous. Additional details of implementation of each command follow in the next section along with examples.

Table 24.2. Key basic TES methods and commands

Regression	RA	<code>teffects ra</code>
IPW	Inverse-probability weighting	<code>teffects ipw</code>
	AIPW	<code>teffects aipw, telasso</code>
	IPW-RA	<code>teffects ipwra</code>
Matching	NNM	<code>teffects nnmatch</code>
	PSM	<code>teffects psmatch</code>
	Multiple treatments	<code>teffects multivalued</code>
Diagnostics	Overlap plots	<code>teoverlap</code>
	Covariate balance	<code>tebalance</code>

In this chapter, we focus on a continuous outcome, though the outcome can also be a count, a binary outcome, or an ordered discrete outcome. For binary or ordered discrete outcomes, the TE is measured as the change in probability due to treatment.

An example of the `teffects` commands is the `teffects ipwra` command, which includes both RA and inverse-probability weighting. The command

syntax is

```
teffects ipwra (ovar omvarlist [ , omodel noconstant ])  
    (tvar tmvarlist [ , tmodel noconstant ]) [if] [in] [weight] [ , stat options ]
```

Here *ovar* is the outcome variable, and *omodel* is the outcome model that can be linear, logit, probit, hetprobit(), poisson, flogit, fprobit, or fhetprobit(). The treatment variable is *tvar*, and *tmodel* can be logit, probit, or hetprobit(). The statistic defined in *stat* can be ate (the default), atet, or pomeans. The vce(*vcetype*) may be robust, cluster *clustvar*, bootstrap, or jackknife. The option aequations displays all equations estimated. Option *pstolerance*(#) sets the tolerance for satisfying the overlap assumption, with default value 10^{-5} , and option *osample()* identifies observations that violate the overlap assumption.

The *teffects ra* command has similar syntax, except that the treatment model is no longer relevant, and the *teffects ipw* command has similar syntax, except that the outcome model is no longer relevant. The *teffects aipw* command has similar syntax, except that the ATET is not identified in this case, and options *nls* and *wnls* provide alternative methods other than the default of maximum likelihood for estimating the conditional means.

The *telasso* command uses the lasso to select a subset of regressors from a wide set of potential regressors in obtaining AIPW estimates; see section [24.6.10](#).

For the *teffects psmatch* command, the outcome model is not relevant. The treatment variable model can be logit, probit, or hetprobit(), and the statistic defined in *stat* can be ate (the default) or atet. The *nneighbor*(#) option specifies the number of matches per observation; the default is one. The *caliper*(#) option defines the maximum distance used to determine potential neighbors, and options *pstolerance*(#) and *osample()* are the same as for the *teffects ipwra* command. The default standard errors (option *vce(robust)*) are heteroskedastic–robust standard errors that allow $\text{Var}(y_{0i}|\mathbf{x}_i)$ and $\text{Var}(y_{1i}|\mathbf{x}_i)$ to vary with covariates and treatment level. These are based on two nearest neighbors; this can be changed using option *vce(robust [, nn(#)])*. The option *vce(iid)* gives homoskedastic standard errors.

The *teffects nnmatch* command has options similar to *teffects psmatch* with the following important variations. The treatment variable model

is no longer relevant. Instead, the metric for matching is specified in the `metric()` option as `mahalanobis` (the default), `ivariance`, `euclidean`, or matrix `matname`. The `ematch(varlist)` option matches exactly on specified variables. `biasadj(varlist)` adjusts for bias that arises if matching is on more than one continuous variable.

Regardless of the estimation method used, it is important to additionally use the `tebalance` and `teoverlap` commands where available to check for covariate balance and propensity-score overlap. We focus on the most common case of a binary treatment. A multivalued treatment example is given in section [24.10](#). In that case, the `teffects` commands aside from the matching estimator commands are available, and an MNL model is used for the propensity scores (option `logit`).

Additional TE commands are `teffects` and `etregress` for linear regression with endogenous treatment, `etpoisson` for Poisson regression with endogenous treatment, and `stteffects` for survival data with exogenous treatment. The last of these commands uses parametric survival models to control for censoring.

24.7.2 ERM commands when treatment is exogenous

Stata's extended regression model (ERM) commands, summarized in section [23.7](#), are focused on parametric models with endogenous regressors and joint normal model errors; see section [25.3](#). These commands cover linear regression (`erregress`), probit regression (`eprobit`), ordered probit (`eoprobit`), and interval regression (`eintreg`).

For discrete-valued treatment with exogenous treatment, the `extreat()` option is used, and the `estat teffects` postestimation command provides the ATE estimate as default; options provide the estimated ATET and the POM. If the model is fit with option `vce(robust)`, then `estat teffects` gives unconditional standard errors that additionally allow for variation in the regressors; see section 13.7.9.

Table [24.3](#) displays alternative ways of doing the same analysis. The left column gives the syntax for the `teffects ra` command, and the right column, for the corresponding ERM command, `extreat()` option, and subsequent `estat`

`teffects` postestimation command. An endogenous treatment application using the `eregress`, `entreat` command is given in section [25.3.3](#).

Table 24.3. `teffects ra` and ERM commands when treatment is exogenous

<code>teffects</code> commands	ERM command
Linear regression with exogenous treatment <code>teffects ra (y x) (D)</code>	<code>eregress y x, extreat(D)</code> <code>estat teffects</code>
Binary probit with exogenous treatment <code>teffects ra (y x, probit) (D)</code>	<code>eprobit y x, extreat(D)</code> <code>estat teffects</code>

24.8 Oregon Health Insurance Experiment example

We use a sample extract from the public use files of the OHIE to illustrate the concepts and methods presented in preceding sections using the relevant Stata commands. The experiment was a lottery whose winners were granted the option of applying for enrollment in the state Medicaid program, a state-run health insurance program that would increase access to healthcare.

In this section, we summarize the experiment and the data analyzed. In the subsequent section, we analyze how the out-of-pocket cost of medical services received varied with whether the individual was a lottery winner.

We compute the ATE of winning the lottery, easily done here because a simple framework of exogenous treatment assignment is appropriate for this RCT. The only complication is that winning the lottery varied with household size and the time of the lottery and the survey, so analysis should control for these variables.

If all lottery winners enrolled in Medicaid, then this ATE would also measure the ATE of being in Medicaid. Not all lottery winners enrolled in Medicaid, however, so the ATE of winning the lottery needs to be viewed as an intention-to-treat effect if the ultimate treatment of interest is enrolling in Medicaid. The more difficult estimation of the ATE of enrolling in Medicaid is deferred to section [25.5](#).

24.8.1 The OHIE

The OHIE is an important modern example of an RCT or a social experiment. The background to this experiment has been covered in detail in [NBER \(n.d.\)](#) and [Baicker et al. \(2013\)](#). Here we provide only the essential details for interpreting the application that follows.

The U.S. Medicaid program, which provides health insurance for low-income people, is run separately by each state. At the time of the experiment, the state of Oregon's Medicaid program, the Oregon Health Program (OHP), was separated into two components. OHP Plus served the categorically eligible Medicaid population, while OHP Standard was an expansion program

targeting low-income uninsured adults who were otherwise ineligible for OHP Plus. Because of budgetary constraints, OHP Standard was wound back and closed to new applicants in 2004, leading to significant attrition over the following four years. Facing an 80% decline in enrollments, in January of 2008, the state determined to expand the program by an additional 10,000 positions.

Importantly for our purposes, the OHP anticipated demand for the program in excess of the available new positions, so it sought and received permission to assign selection by lottery. Such random assignment provides a rare opportunity to assess the impact of expanded health insurance coverage on a variety of health and financial outcomes within RCT design framework.

Indeed, from February to March 2008, some 90,000 individuals registered for the lottery list. Over the next 6 months, the government conducted 8 waves of lottery draws, resulting in some 35,000 individuals being offered the opportunity to apply for OHP coverage. Note that the opportunity to apply was extended to all members of the selected individual's household; thus, selection was random conditional on the number of household members in the lottery list. Approximately 35,000 individuals from 30,000 unique households were selected, and of those approximately 30% were eligible and enrolled in Medicaid by the given deadlines.

Following the treatment, researchers tracked lottery participant outcomes over the next 12 months with 3 mail surveys. We analyze data from the third of these mail surveys, which was undertaken in 7 mail-out waves approximately 12 months after the lottery (July and August 2009). Nearly all individuals selected in the lottery, as well as an approximately equal number of nonselected individuals, were mailed questionnaires regarding healthcare needs, experiences, and costs over the previous six months. Following an intensive follow-up protocol undertaken on a subset of nonresponders, the researchers achieved an estimated response rate of approximately 50%.

In this chapter, we view the treatment as being selected by the lottery to be eligible for Medicaid. We include indicator variables capturing household size and survey wave to control for potential correlation with the probability

of treatment. And we include a set of relevant covariates to improve efficiency, including variables relating to smoking status, income as a percentage of the federal poverty line, education level, and employment.

Not all individuals selected by the lottery actually enrolled in Medicaid. In section [25.5](#), we view the treatment to be actual enrollment in Medicaid and obtain a local average TE estimate that uses selection by the lottery as an instrument for actual enrollment in Medicaid. In that context, the ATE that we estimate in the analysis below is called an intention-to-treat effect.

24.8.2 Data summary

The size and complexity of the OHIE dataset is reflected in data files in [NBER \(n.d.\)](#). Given our limited objective of illustrating the methods in this chapter, rather than carrying out a full-scale empirical study, we focus on a single outcome variable and limit the number and type of exogenous regressors that are used.

The key data can be grouped into outcomes, treatments, and regressors, with the global `zlist` referring to household variables and wave variables that may affect lottery participation and the global `xlist` referring to individual-specific variables that may affect the outcome variable.

```
. * Variables: (1) outcomes (2) z: treatment related (3) x: outcome related
. qui use mus224ohiesmallrecode, clear
. global outcomes oop dowe ervisits // Outcome variables
. global y oop // Outcome variable for this chapter
. global hh dhssize2 dhssize3 // Household size dummies
. global wave dlotdraw* dsurvdraw* // Lottery and survey draws
. global zlist $hh $wave // z variables for the treatment
. global xlist dsmoke hhinc deduc2-deduc4 demploy2-demploy4 // x for outcome
```

A brief description of the variables used in the following analysis is as follows.

```
. * Variable descriptions
. describe $outcomes lottery medicaid household_id $zlist $xlist
```

Variable name	Storage type	Display format	Value label	Variable label
oop	float	%9.0g		Out of pocket costs (cost_tot_oop_mod_12m)
dowe	byte	%9.0g	yesno	Owe any money for health (cost_any_owe_12m owe_any)
ervisits	byte	%9.0g		Emergency room visits (er_num_mod_12m)
lottery	byte	%12.0g	lottery	Selected in the lottery
medicaid	byte	%12.0g	enrolled	Ever enrolled in Medicaid from 1st not. date (10mar2008) to 30sep2009 (ohp_all_e)
household_id	float	%9.0g		Scrambled household identifier
dhhsiz2	byte	%8.0g		2 in hh (dddnumhh_li_2)
dhhsiz3	byte	%8.0g		3 in hh (dddnumhh_li_3)
dlotdraw2	byte	%8.0g		draw_lottery==2 (llldraw_lot_2)
dlotdraw3	byte	%8.0g		draw_lottery==3 (llldraw_lot_3)
dlotdraw4	byte	%8.0g		draw_lottery==4 (llldraw_lot_4)
dlotdraw5	byte	%8.0g		draw_lottery==5 (llldraw_lot_5)
dlotdraw6	byte	%8.0g		draw_lottery==6 (llldraw_lot_6)
dlotdraw7	byte	%8.0g		draw_lottery==7 (llldraw_lot_7)
dlotdraw8	byte	%8.0g		draw_lottery==8 (llldraw_lot_8)
dsurvdraw2	byte	%8.0g		draw_survey==2 (ddddraw_sur_2)
dsurvdraw3	byte	%8.0g		draw_survey==3 (ddddraw_sur_3)
dsurvdraw4	byte	%8.0g		draw_survey==4 (ddddraw_sur_4)
dsurvdraw5	byte	%8.0g		draw_survey==5 (ddddraw_sur_5)
dsurvdraw6	byte	%8.0g		draw_survey==6 (ddddraw_sur_6)
dsurvdraw7	byte	%8.0g		draw_survey==7 (ddddraw_sur_7)
dsmoke	byte	%10.0g	smk_lbl	Currently smoke cigs (smk_curr_12m)
hhinc	float	%3.0f		Household income as % of federal poverty line (hhinc_pctfp1_12m)
deduc2	byte	%8.0g		HS diploma or GED (edu_12m_2)
deduc3	byte	%8.0g		Voc or 2yr degree (edu_12m_3)
deduc4	byte	%8.0g		Four year degree (edu_12m_4)
demploy2	byte	%8.0g		Work < 20 hrs/wk (employ_hrs_12m_2)
demploy3	byte	%8.0g		Work 20-29 hrs/wk (employ_hrs_12m_3)
demploy4	byte	%8.0g		Work 30+ hrs/wk (employ_hrs_12m_4)

The names in parentheses are the original names of the variables in the OHIE dataset. The suffix `_12m` refers to tracking of lottery participant outcomes over the next 12 months; the survey questions in fact covered healthcare needs, experiences, and costs over the previous 6 months.

The continuous outcome variable analyzed is out-of-pocket expenditures (`oop`) in the past six months. Additional outcomes that we do not analyze, reserving them for exercises, are a binary outcome of whether one owes any money (`dowe`) and a count outcome of emergency room visits (`ervisits`).

The treatment considered in this chapter is whether one wins the lottery (lottery), while the next chapter analyzes enrollment in Medicaid (medicaid).

- . * Summary statistics: Outcomes, treatments, and household size
- . summarize \$outcomes lottery medicaid \$hh

Variable	Obs	Mean	Std. dev.	Min	Max
oop	22,679	269.0062	733.0821	0	9400
dowe	22,476	.5542801	.497056	0	1
ervisits	22,491	.4372416	.9812417	0	10
lottery	22,679	.4972001	.5000032	0	1
medicaid	22,679	.2812293	.4496091	0	1
dhssize2	22,679	.2963094	.4566392	0	1
dhssize3	22,679	.0025133	.0500713	0	1

While 49.7% of lottery participants won the lottery, the treatment of this chapter, only 28.1% of the sample actually enrolled in Medicaid, a treatment analyzed in chapter [25](#).

Although the state randomly sampled from individuals on the list, the entire household of any selected individual was considered selected and eligible to apply for OHP Standard. Thus, selected (treatment) individuals are disproportionately drawn from households of larger household size. Additionally, for the sample at hand, winning the lottery varied with the time of the lottery and the survey, as the following linear probability model indicates.

```

. * Lottery dummy varies with household size and lottery and survey waves
. regress lottery $zlist, vce(cluster household_id)

Linear regression                                         Number of obs      =     22,679
                                                               F(15, 20147)      =      93.29
                                                               Prob > F        =     0.0000
                                                               R-squared       =     0.0677
                                                               Root MSE        =     .48294

(Std. err. adjusted for 20,148 clusters in household_id)

```

lottery	Coefficient	Robust				
		std. err.	t	P> t	[95% conf. interval]	
dhhsiz2	.0651882	.0088219	7.39	0.000	.0478965	.0824798
dhhsiz3	.2807319	.0618745	4.54	0.000	.1594529	.402011
dlotdraw2	.002456	.0168842	0.15	0.884	-.0306384	.0355504
dlotdraw3	.0006732	.016864	0.04	0.968	-.0323816	.0337279
dlotdraw4	.0541089	.0164205	3.30	0.001	.0219234	.0862944
dlotdraw5	.0747288	.016741	4.46	0.000	.0419149	.1075426
dlotdraw6	.0850831	.0144081	5.91	0.000	.056842	.1133243
dlotdraw7	.0959815	.0144576	6.64	0.000	.0676434	.1243197
dlotdraw8	.0855691	.0165617	5.17	0.000	.0531068	.1180315
dsurvdraw2	.0122423	.0188461	0.65	0.516	-.0246976	.0491822
dsurvdraw3	.0205117	.0188443	1.09	0.276	-.0164247	.0574481
dsurvdraw4	-.1936876	.0163184	-11.87	0.000	-.225673	-.1617021
dsurvdraw5	-.2042628	.0162951	-12.54	0.000	-.2362026	-.1723231
dsurvdraw6	-.2686614	.0149923	-17.92	0.000	-.2980475	-.2392753
dsurvdraw7	-.3149375	.0148982	-21.14	0.000	-.3441391	-.2857359
_cons	.5820885	.011587	50.24	0.000	.559377	.6048001

Separate analysis of each variable shows that lottery success by household size ranged from 0.57 to 0.89, by lottery draw ranged from 0.48 to 0.53, and by survey ranged from 0.34 to 0.66. Given this, we expect fitted propensity scores to be approximately in the range 0.3 to 0.9. We use the variables in `zlist` as controls in the analysis below. [Finkelstein et al. \(2012\)](#) used a richer list that added interactions of the wave variables with the household size indicator variables.

To illustrate the use of control variables that may increase the efficiency of estimators of an outcome model, the analysis below uses an indicator for smoker (`dsmoke`), household income (`hhinc`), and education and employment indicator variables.

```
. * Summary statistics: xlist variables that may improve outcome model fit
. summarize $xlist
```

Variable	Obs	Mean	Std. dev.	Min	Max
dsmoke	22,154	2.262661	.9171565	1	3
hhinc	20,478	76.97273	69.16905	0	461.6898
deduc2	21,986	.4982716	.5000084	0	1
deduc3	21,986	.2204585	.4145653	0	1
deduc4	21,986	.1137997	.317575	0	1
demploy2	22,411	.0912052	.2879071	0	1
demploy3	22,411	.0999509	.2999412	0	1
demploy4	22,411	.2638436	.4407254	0	1

24.8.3 Initial regression analysis

Before demonstrating the various `teffects` commands, we perform some initial analysis.

The following quantile plots truncate `oop` at \$2,000 (the 97th percentile) for readability.

```
. * Quantile plot of outcome variable by treatment status
. qui qplot $y if $y < 2000, over(lottery) clpattern(1 _) recast(line)
>      ytitle("Out-of-pocket spending") legend(pos(11) ring(0) col(1))
>      title("Quantile plots for lottery") lwidth(medthick thick)
>      xtitle("Fraction of the data")
>          lpattern(solid dash)
. qui qplot $y if $y < 2000, over(medicaid) clpattern(1 _) recast(line)
>      ytitle("Out-of-pocket spending") legend(pos(11) ring(0) col(1))
>      title("Quantile plots for Medicaid") lwidth(medthick thick)
>      xtitle("Fraction of the data")
>          lpattern(solid dash)
```

The first panel of figure [24.1](#) is very similar to the plot given in [Finkelstein et al. \(2012, 1093\)](#). About half the sample had zero out-of-pocket expenditures. In principle, the outcome could be modeled using a two-part model; instead, an RA uses the linear model. The out-of-pocket expenditures are lower for lottery winners. The difference is even greater for those who enroll in Medicaid; this is studied in chapter [25](#).

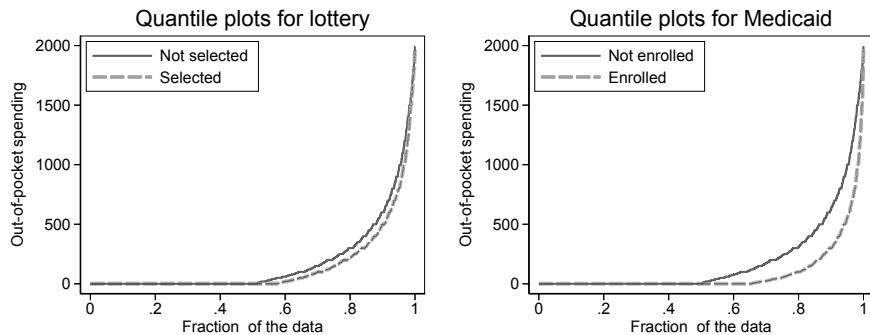


Figure 24.1. Quantile plots of TE

A starting point is OLS regression of the outcome on a treatment dummy and some additional control variables, where in the most general case,

$$y_{ih} = \beta_0 + \beta_1 \text{treat}_h + \mathbf{z}'_{ih} \boldsymbol{\beta}_2 + \mathbf{x}'_{ih} \boldsymbol{\beta}_3 + \varepsilon_{ih}$$

where i is an individual subscript, h is the household subscript, \mathbf{z} are control variables for the treatment assignment, necessary to ensure that assumption 1 in section [24.5](#) is satisfied, and \mathbf{x} are control variables related to the outcome that may improve the precision of estimates.

We fit OLS regressions of `oop` on `lottery` with increasing sets of controls.

- . * Regression of outcome on treatment with various controls
- . qui regress \$y lottery, vce(robust)
- . estimates store Diff_rob
- . qui regress \$y lottery, vce(cluster household_id)
- . estimates store Diff_clu
- . qui regress \$y lottery \$zlist, vce(cluster household_id)
- . estimates store zlist
- . qui regress \$y lottery \$xlist, vce(cluster household_id)
- . estimates store xlist
- . qui regress \$y lottery \$zlist \$xlist, vce(cluster household_id)
- . estimates store Both

```
. estimates table Diff_rob Diff_clu zlist xlist Both, keep(lottery)
> b(%8.4f) se stat(N r2)
```

Variable	Diff_rob	Diff_clu	zlist	xlist	Both
lottery	-44.6627 9.7294	-44.6627 9.9216	-40.9243 10.1302	-45.7420 10.6484	-40.1693 10.8169
N	22679	22679	22679	19393	19393
r2	0.0009	0.0009	0.0018	0.0147	0.0154

Legend: b/se

The results are consistent across the regressions, with a highly statistically significant reduction of \$40–\$45 on a base of \$270. As already explained, the preferred estimates include the *z* variables as controls.

The first estimates are essentially the same as those from `ttest oop, by(lottery) unequal`. The second estimates use cluster-robust standard errors that cluster on household, something done in all the subsequent analysis, aside from matching estimators, leading to slightly higher standard errors. The third estimates add various lottery and survey variables as controls. The fourth estimates included additional variables likely to be correlated with the outcome, leading to a loss of 3,286 observations due to missing data. The fifth combines both sets of controls. The control variables have very low explanatory power.

24.9 Treatment-effect estimates using the OHIE data

The article by [Finkelstein et al. \(2012\)](#), 1093) and the supplemental material provide an excellent benchmark for evaluation of an RCT. The impact of `lottery` on various outcomes was estimated by regression of the outcome on `lottery`, and an expanded list of the lottery variables given in `zlist`. Because this was a well-designed RCT, more complex analysis was unnecessary.

For pedagogical purposes, we demonstrate the more complex analysis. This becomes essential when data are from an observational study, rather than an RCT, though is valid in an observational study only if sufficient controls are included to make the assumption of selection on observables only a reasonable assumption.

We consider in turn regression imputation methods, IPW methods, and matching methods. For brevity, we generally estimate only the ATE, the default, and use default computer output that simply provides the ATE. The options `atet` and `pomeans` provide estimates of the ATET and POMS, and the `aequations` option lists the estimates of the underlying regression models.

For this RCT, we expect the various methods to lead to an estimated ATE similar to that from simple OLS regression on `lottery` and the `z` variables. In other applications with observational data, we expect bigger departures from simple OLS regression, though in an ideal world, we expect similar answers across the various treatment evaluation methods.

24.9.1 RA estimates

We perform RA analysis (see section [24.6.1](#)), with controls both the necessary treatment assignment controls (`zlist`) and additional controls that may increase estimator precision (`xlist`). The default `teffects ra` command provides the estimated ATE. The option `aequations` additionally provides the estimates of the two underlying OLS regressions.

```

. * Regression-adjusted ATE using $zlist and $xlist
. teffects ra ($y $xlist $zlist) (lottery), nolog vce(cluster household_id)
Treatment-effects estimation                               Number of obs      =    19,393
Estimator        : regression adjustment
Outcome model   : linear
Treatment model: none
                                         (Std. err. adjusted for 17,348 clusters in household_id)

```

oop	Robust					[95% conf. interval]
	Coefficient	std. err.	z	P> z		
ATE lottery (Selected vs Not selected)	-40.44588	10.94579	-3.70	0.000	-61.89924	-18.99252
P0mean lottery Not selected	292.1394	7.693716	37.97	0.000	277.06	307.2188

The ATE estimate and its standard error are very similar to the estimates of -40.17 and 10.82 obtained earlier by the simpler single OLS regression.

For completeness, we also obtain the estimated POMS.

```

. * Regression-adjusted POMs using $zlist and $xlist
. teffects ra ($y $xlist $zlist) (lottery), pomeans nolog vce(clu household_id)
Treatment-effects estimation                               Number of obs      =    19,393
Estimator        : regression adjustment
Outcome model   : linear
Treatment model: none
                                         (Std. err. adjusted for 17,348 clusters in household_id)

```

oop	Robust					[95% conf. interval]
	Coefficient	std. err.	z	P> z		
P0means lottery Not selected	292.1394	7.693716	37.97	0.000	277.06	307.2188
Selected	251.6935	7.831481	32.14	0.000	236.3441	267.0429

The difference $251.6935 - 292.1394 = -40.4459$ is the preceding ATE estimate. By comparison, the raw means of the data by lottery status for this sample with $N = 19393$ equal 291.2125 and 246.5498 .

We next obtain the ATET.

```

. * Regression-adjusted ATET using $zlist and $xlist
. teffects ra ($y $xlist $zlist) (lottery), atet nolog vce(cluster household_id)
Treatment-effects estimation                                         Number of obs      =    19,393
Estimator          : regression adjustment
Outcome model     : linear
Treatment model: none
                                                (Std. err. adjusted for 17,348 clusters in household_id)

```

oop	Robust				
	Coefficient	std. err.	z	P> z	[95% conf. interval]
ATET lottery (Selected vs Not selected)	-35.98451	11.10953	-3.24	0.001	-57.75878 -14.21024
P0mean lottery Not selected	288.2787	8.271055	34.85	0.000	272.0678 304.4897

The estimated ATET is – 35.98 compared with the ATE estimate of – 40.45. The difference between the two reflects in part the particular composition of the lottery winner group compared with all lottery participants. Significant differences in the impact of control variables on the respective outcomes of the treated and untreated may also partially account for difference between the two TES.

24.9.2 IPW estimates

The IPW method is presented in section [24.6.2](#). The default `teffects ipw` command obtains weights based on a logit model. Here the regressors are `zlist` variables associated with the randomization design, so the sample size is larger than the preceding example where `xlist` variables were also used.

```

. * IPW ATE using $zlist
. teffects ipw ($y) (lottery $zlist), nolog vce(cluster household_id)
Treatment-effects estimation                               Number of obs      =     22,679
Estimator        : inverse-probability weights
Outcome model   : weighted mean
Treatment model: logit
                                         (Std. err. adjusted for 20,148 clusters in household_id)

```

oop	Robust				
	Coefficient	std. err.	z	P> z	[95% conf. interval]
ATE lottery (Selected vs Not selected)	-39.56951	10.12338	-3.91	0.000	-59.41098 -19.72804
P0mean lottery Not selected	286.0328	7.275387	39.32	0.000	271.7733 300.2923

The ATE estimate of -39.57 is similar to the estimates from the regression-adjusted model, despite a difference in sample size, and the confidence interval $[-59.4, -19.7]$ is slightly narrower because of a lower standard error.

24.9.3 Propensity-score overlap

The propensity score should be evaluated for two reasons. First, an important assumption is the overlap assumption that for each value of the control variables, there are both treated and untreated cases. Second, if the propensity score for an observation is very close to 0 or very close to 1, then IPW methods can give that observation a very high weight.

The postestimation command `teoverlap` provides a kernel density estimate plot for the propensity score at the different values of the treatment variable. Ideally, the two densities are very similar. The default is to calculate predicted probabilities for the lowest treatment level, here $D = 0$. We add the option `ptlevel(1)` so that the predicted probabilities are instead those of the propensity score, $\Pr(D = 1|\mathbf{z})$. We have

```

. * Overlap assumption - graph using teoverlap
. teoverlap, ptlevel(1) kernel(triangle) bw(0.04)
>      title("Kernel density overlap") xtitle("Propensity score")

```

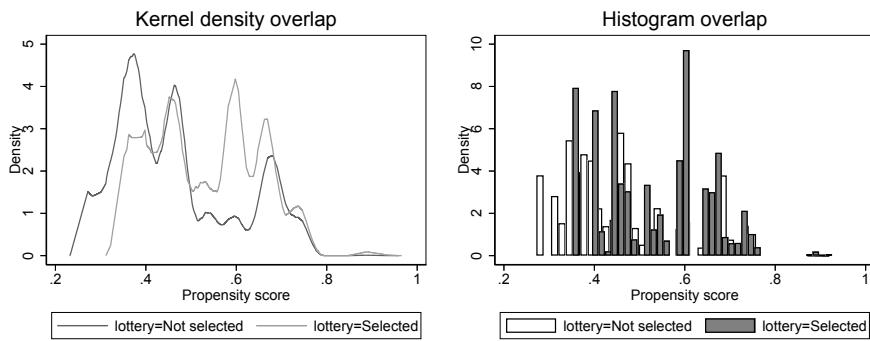


Figure 24.2. Propensity-score overlap

The first panel of figure 24.2 plots the two propensity-score densities. The propensity scores appear to be mostly in the range 0.25 to 0.75, so they are away from the boundaries of 0 and 1. For lottery winners, the lighter line, the lowest propensity score is only 0.3, yet many lottery losers, the darker line, have a lower propensity score. And there is a slight bump for lottery winners around 0.9. The plots suggest that the overlap is essentially in the range 0.3 to 0.75. Analysis might best be restricted to that range, leading to a loss of 872 observations or 3.8% of the sample, or at least restricted to that range in a robustness test.

The `predict` postestimation command following the `teffects ipw` command computes the predicted probability that $D = 0$, so the propensity score can be computed as one minus this estimate. Equivalently, we predict following the `logit` command.

```
. * Overlap assumption - manual graph using logit prediction and histograms
. qui logit lottery $zlist, vce(cluster household_id)
. qui predict lothat
. summarize lothat
    Variable |       Obs        Mean      Std. dev.       Min       Max
    lothat |   22,679     .4972001     .1301477     .2716641     .9257978
. twoway (hist lothat if lottery==0, fcol(white)) (hist lothat if lottery==1),
>         title("Histogram overlap") xtitle("Propensity score")
>         legend(label(1 "lottery=Not selected") label(2 "lottery=Selected"))
```

The IPW weights will not take extreme values, even if all the sample is used, because the propensity scores range from 0.272 to 0.926. This range does

not really change if a more flexible model is used by fully interacting the variables in `zlist`. As explained in section [24.8](#), for this RCT example, propensity scores are expected to be a considerable amount away from the boundary values of 0 and 1. For an observational study with a very flexible binary outcome, model extreme values may be more likely to occur.

The second panel of figure [24.2](#) plots histograms of the propensity scores by lottery status and leads to the same conclusions as the first panel. If propensity scores are felt to not overlap or are felt to be too close to 0 or 1, leading to concerns about IPW estimator instability, one can trim using the estimated propensity scores and the `if` qualifier. For example, give command `teffects ipw ($y) (lottery $zlist) if (lothat>0.3) & lothat<0.75`.

24.9.4 Regressor balance

Regressor balance is essential. The `tebalance` postestimation commands are available after all `teffects` commands that use propensity scores or matching but are not available after the `teffects ra` command.

Here we check balance following the `teffects ipw` command, beginning with the `tebalance summarize` command.

```

. * Covariate balance summary
. qui teffects ipw ($y) (lottery $zlist), nolog vce(cluster household_id)
. tebalance summarize
Covariate balance summary

```

	Raw	Weighted
Number of obs =	22,679	22,679.0
Treated obs =	11,276	11,151.4
Control obs =	11,403	11,527.6

	Standardized differences		Variance ratio	
	Raw	Weighted	Raw	Weighted
dhhsize2	.1939117	.0060577	1.19006	1.005543
dhhsize3	.0797051	.0021402	8.56137	1.04358
dlotdraw2	.0331758	.0015502	1.089294	1.004479
dlotdraw3	.0365961	.0046157	1.100489	1.013669
dlotdraw4	-.0257401	-.0069954	.9342648	.9817981
dlotdraw5	.0003948	-.0069064	1.001076	.9816374
dlotdraw6	-.0083996	-.0028377	.9873158	.9960122
dlotdraw7	-.0192988	.0054252	.970481	1.007623
dlotdraw8	-.0254451	.0079989	.9341439	1.020275
dsurvdraw2	.2260162	-.0055742	1.772475	.9868448
dsurvdraw3	.2397501	-.0094081	1.838261	.9780227
dsurvdraw4	-.0404115	.0106734	.9196795	1.022566
dsurvdraw5	-.0458372	.0091592	.9098577	1.019404
dsurvdraw6	-.1878871	.0052308	.7566851	1.007905
dsurvdraw7	-.290524	-.0054474	.5950349	.9903239

The first part of the output shows that after weighting, the effective number of treated and control observations is similar to that in the raw data. In other settings, especially with observational data, there can be much bigger differences.

The output then shows both the raw and weighted differences that were defined in section [24.6.4](#). The raw difference measure, calculated variable by variable, refers to differences before applying the inverse-probability weights, while weighted differences refer to the differences after applying inverse-probability weighting. The weighted differences are small, and most of the variance ratios are close to 1, indicating that weighting has corrected the lack of balance in the raw data, especially that in variables `dhhsize3`, `dsurvdraw6`, and `dsurvdraw7`.

The `tebalance density` command provides diagnostic plots for individual variables that contrast a kernel density plot for the raw variable with the kernel density plot for the weighted variable.

Following the `teffects nnmatch` and `teffects psmatch` commands, the `tebalance box` command provides diagnostic box plots for individual variables.

Following the `teffects ipw`, `teffects aipw`, and `teffects ipwra` commands, the `tebalance overid` command provides a formal test of restrictions imposed by the balance requirement. For the current example, we obtain

```
. * Formal test of balancing
. tebalance overid
Iteration 0: criterion = .00024449
(output omitted)
Iteration 192: criterion = .00278518 (backed up)
Overidentification test for covariate balance
H0: Covariates are balanced:
chi2(16)      = 358.044
Prob > chi2   = 0.0000
```

The null hypothesis of balance is rejected, though in general, with a very large sample size, almost any specification test is likely to reject at a fixed level of significance such as 0.05. The reasonable balance after weighting suggested by the `tebalance summary` command may be a better guide. We revisit balance in section [24.9.7](#).

24.9.5 AIPW and IPW-RA estimates

We next present the two doubly robust hybrid methods detailed in section [24.6.5](#) that combine RA and inverse-probability weighting.

For the AIPW estimator, we add the `aequations` option to the `teffects aipw` command to obtain output that in addition to the estimated ATE includes the estimates for the two regressions for the POM and the logit regression for the propensity score.

```

. * Augmented IPW ATE
. teffects aipw ($y $xlist) (lottery $zlist), aequations nolog
> vce(cluster household_id)

Treatment-effects estimation                               Number of obs      =    19,393
Estimator        : augmented IPW
Outcome model   : linear by ML
Treatment model: logit
                                         (Std. err. adjusted for 17,348 clusters in household_id)

```

oop	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
ATE						
lottery (Selected vs Not selected)	-38.66964	10.79616	-3.58	0.000	-59.82973	-17.50956
P0mean						
lottery Not selected	290.2338	7.720553	37.59	0.000	275.1018	305.3658
OME0						
dsmoke	.9205123	8.662252	0.11	0.915	-16.05719	17.89821
hhinc	1.005393	.1398412	7.19	0.000	.731309	1.279477
deduc2	8.705563	21.02138	0.41	0.679	-32.49559	49.90671
deduc3	78.9117	25.07941	3.15	0.002	29.75696	128.0664
deduc4	61.82869	29.22027	2.12	0.034	4.558018	119.0994
demploy2	-16.27391	26.17013	-0.62	0.534	-67.56643	35.01861
demploy3	-27.16429	24.70588	-1.10	0.272	-75.58692	21.25834
demploy4	-27.29717	20.0837	-1.36	0.174	-66.66049	12.06615
_cons	199.0383	25.9641	7.67	0.000	148.1496	249.927
OME1						
dsmoke	23.37641	8.211131	2.85	0.004	7.282887	39.46993
hhinc	1.1747	.1409721	8.33	0.000	.8983998	1.451
deduc2	-26.81169	22.21343	-1.21	0.227	-70.34921	16.72582
deduc3	38.89857	26.51898	1.47	0.142	-13.07768	90.87481
deduc4	17.02745	33.1716	0.51	0.608	-47.98769	82.04259
demploy2	-59.06455	21.31782	-2.77	0.006	-100.8467	-17.2824
demploy3	-74.72525	21.30281	-3.51	0.000	-116.478	-32.97251
demploy4	-18.49857	20.74892	-0.89	0.373	-59.1657	22.16856
_cons	127.8238	25.06627	5.10	0.000	78.69479	176.9528

TME1							
dhsiz2	.2712528	.0397604	6.82	0.000	.1933239	.3491817	
dhsiz3	1.671956	.699984	2.39	0.017	.300013	3.0439	
dlotdraw2	.0329293	.0817694	0.40	0.687	-.1273357	.1931943	
dlotdraw3	-.0134305	.0800556	-0.17	0.867	-.1703366	.1434756	
dlotdraw4	.2419665	.077133	3.14	0.002	.0907886	.3931443	
dlotdraw5	.3180178	.0781217	4.07	0.000	.1649021	.4711335	
dlotdraw6	.3624306	.0681201	5.32	0.000	.2289176	.4959436	
dlotdraw7	.437767	.0687437	6.37	0.000	.3030318	.5725023	
dlotdraw8	.3686761	.0777056	4.74	0.000	.216376	.5209762	
dsurvdraw2	.0720634	.093083	0.77	0.439	-.110376	.2545028	
dsurvdraw3	.0894928	.0923688	0.97	0.333	-.0915468	.2705323	
dsurvdraw4	-.8057285	.0764003	-10.55	0.000	-.9554703	-.6559866	
dsurvdraw5	-.8464725	.0764515	-11.07	0.000	-.9963147	-.6966302	
dsurvdraw6	-1.117297	.0717738	-15.57	0.000	-1.257971	-.9766228	
dsurvdraw7	-1.319487	.0724789	-18.21	0.000	-1.461543	-1.177431	
_cons	.3340067	.0534825	6.25	0.000	.2291829	.4388306	

The ATE estimate of -38.67 is slightly lower than that from the `ipw` output.

IPW-RA using defaults for the `teffects ipwra` command yields the following result:

```
. * IPW with RA ATE
. teffects ipwra ($y $xlist $zlist) (lottery $zlist), nolog
> vce(cluster household_id)

Treatment-effects estimation                               Number of obs      =    19,393
Estimator        : IPW regression adjustment
Outcome model   : linear
Treatment model: logit
                                         (Std. err. adjusted for 17,348 clusters in household_id)
```

oop	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
ATE						
lottery (Selected vs Not selected)	-39.93113	10.93502	-3.65	0.000	-61.36337	-18.49888
P0mean						
lottery Not selected	291.8993	7.686289	37.98	0.000	276.8344	306.9641

The ATE estimate of -39.93 is very similar to -38.67 obtained using AIPW.

The AIPW can also be obtained using the `telasso` command, which used the lasso to select a subset of many potential control variables; see section [28.9.4](#).

24.9.6 PSM estimates

The `teffects psmatch` command implements PSM; see section [24.6.7](#). The matches are created using Stata's default option. Default heteroskedastic-robust standard errors are used because obtaining standard errors for matching estimators is a nonstandard inference problem, and the `vce(cluster clustvar)` option is unavailable for matching estimators.

* PSM ATE					
. * PSM ATE					
. teffects psmatch (\$y) (lottery \$zlist)					
Treatment-effects estimation		Number of obs = 22,679			
Estimator : propensity-score matching		Matches: requested = 1			
Outcome model : matching		min = 1			
Treatment model: logit		max = 1018			
		AI robust			
oop		Coefficient	std. err.	z	P> z [95% conf. interval]
ATE					
lottery (Selected vs Not selected)		-48.935	14.08062	-3.48	0.001 -76.53251 -21.33749

The resulting ATE estimate of -48.94 is about 20% larger in absolute value than the preceding estimates.

24.9.7 Blocking and stratified estimate

As already noted, the `teoverlap` graph clearly suggests that overlap is not uniformly good over the range of propensity scores. This provides added incentive to use blocking, presented in section [24.6.5](#). We use the community-contributed `pscore` command ([Becker and Ichino 2002](#)) because the `teffects` commands do not provide an option for blocking. The `pscore` command provides a detailed balancing test by first blocking the observations by ranges of the propensity score and then testing for regressor balance within each block. The Stata `tebalance` command instead uses all

observations, though one could manually form blocks and apply the `tebalance` command to each block.

The `pscore` command algorithm proceeds as follows. The propensity score is estimated using a probit (the default) or logit model, and the fitted propensity-score values are sorted in ascending order. The sample is split into k equally spaced intervals (blocks) of the ordered propensity score, where k takes default value 5 or a user-specified value (here 8). Within each interval, we test that the average propensity score of treated and control units does not differ and split the interval into two if the test fails in that interval and test again. The procedure stops when in all intervals the average propensity score of treated and control units does not differ. Separate balancing tests of equality between treated and control units are performed for each regressor within each block.

We use the `blockid()` and `pscore()` options to save results for subsequent estimation of the ATE. We suppress the lengthy output, which includes

```
. * Blocking and subsequent balancing tests using community-contributed pscore  
> command  
. pscore lottery $zlist, logit blockid(myblock) pscore(pshat) numblo(8)  
    (output omitted)
```

The final number of blocks is 43

This number of blocks ensures that the mean propensity score
is not different for treated and controls in each blocks
(output omitted)

The balancing property is not satisfied

Try a different specification of the propensity score
(output omitted)

From output not reproduced here, there are 38 blocks that range in size from 57 to 2,830 observations. Of the 550 distinct block-regressor combinations, 25 are unbalanced, leading to the warning message.

The `pscore` package includes commands that implement various matching methods; some are not covered by the `teffects` matching commands. It also includes the `atts` command, which obtains a stratified

estimate of ATE that is a weighted average of the ATES in each block. We obtain

```
. * PSM ATE using blocking and community-contributed atts command
. atts oop lottery, blockid(myblock) pscore(pshat)
```

ATT estimation with the Stratification method
Analytical standard errors

n. treat.	n. contr.	ATT	Std. Err.	t
11276	11403	-34.860	13.552	-2.572

The estimated ATE is -34.86 , a little lower in absolute value than the preceding estimates, which ranged from -38.67 to -48.94 .

24.9.8 NNM estimates

For NNM, detailed in section [24.6.9](#), we split the variables in `zlist` into the `wave` variables, for which matching is based on Mahalanobis distance, and household size, for which matching is exact. Because only a few households had more than two members, we combine those households with two-member households. The `teffects nnmatch` command yields

Treatment-effects estimation		Number of obs = 22,679		
Estimator : nearest-neighbor matching		Matches: requested = 1		
Outcome model : matching		min = 5		
Distance metric: Mahalanobis		max = 1018		
oop		AI robust Coefficient	std. err.	z P> z [95% conf. interval]
ATE lottery (Selected vs Not selected)		-48.68898	13.83561	-3.52 0.000 -75.80629 -21.57168

The estimated ATE is -48.69 , compared with the preceding estimates, which ranged from -38.67 to -48.94 .

24.10 Multilevel treatment effects

The preceding discussion has focused on a binary setup with just one level of treatment that is either received or not received. Forcing treatment evaluation into a binary framework may lead to aggregation of a collection of heterogeneous treatments. For example, contrasting the uninsured with the insured population would be an oversimplification if there are many types of insurance plans that are heterogeneous with respect to coverage, premiums, deductibles, benefits, and so forth. A multilevel treatment (MLT) model allows greater homogeneity of treatment within each level of treatment.

In a multilevel model, each subject receives one of several mutually exclusive treatments. The treatment levels may be ordered according to some criterion. The treatments may be exogenous or self-selected. For example, in a double-blind drug trial, there may be several alternative dosages, and each level of intensity can be considered a separate treatment. In a job training setting, the participants may be subjected to or choose between training periods of different lengths and intensity. TEs then refer to the impact of the treatment relative to not receiving any treatment or receiving one of the other treatments in the available set.

24.10.1 Multivariate TEs methods

We extend the RA approach of binary treatment to the multilevel case. Assume that there are m mutually exclusive treatments and one of the alternatives is the base level. Regardless of whether treatments are ordered or unordered, there can be pairwise comparisons between adjacent levels of treatment or even between any other pair; in each case, the base level can change.

An RA regression model for the outcome y is

$$y_{ij} = \alpha_j D_{ij} + \mathbf{x}'_i \boldsymbol{\beta}_j + \varepsilon_{ij}, \quad i = 1, \dots, N_j, \quad j = 1, \dots, m$$

where D_{ij} denotes the binary indicator for the j th treatment and \mathbf{x} does not include an intercept.

The assumptions regarding treatment assignment are those that were used in the binary case. We assume individualistic treatment, probabilistic treatment, and conditional independence. Hence, we again make the selection-on-observables assumption that $E(\varepsilon_{ij}|\mathbf{x}_i, D_{ij}) = 0$.

The objective is the estimation of TES based on a paired comparison of the potential outcomes of those who are assigned treatment j , the treated group, with those assigned to treatment k , the control group. That is, the ATE for the (j, k) th pair is

$$\begin{aligned}\text{ATE}(j, k) &= E(y_{ij}|\mathbf{x}_i, D_{ij} = 1) - E(y_{ik}|\mathbf{x}_i, D_{ik} = 1) \\ &= \mathbf{x}'_i(\boldsymbol{\beta}_j - \boldsymbol{\beta}_k) + (\alpha_j - \alpha_k)\end{aligned}$$

An immediate implication is that MLT involves more parameters, more counterfactuals, and more computations to support pairwise comparisons. The binary treatment methodology can still be used for specified pairwise comparisons. However, in general, the MLT framework will deliver greater efficiency; see [Cattaneo \(2010\)](#), who establishes consistency and normality of a class of ATE estimators.

Stata's `teffects` commands can be used to model MLT data. The commands `teffects ra`, `teffects ipw`, `teffects ipwra`, and `teffects aipw` extend to MLTs and are illustrated in the next section. Currently, the `teffects psmatch` and `teffects nnmatch` commands are not available for MLT.

24.10.2 Multivalued TEs application

For illustration, we use data on annual prescription drug expenditures of the elderly Medicare population in the United States in 2003 and 2004 derived from the Medicare Current Beneficiary Survey. At that time, Medicare did not cover prescription drug expenditures, unless one was in the hospital.

The data used here are a subset of the data used in [Li and Trivedi \(2016\)](#). Prescription drug coverage can be obtained through various privately

obtained sources, including Medicare supplemental insurance plans (`Medigap`), Medicare managed care plans (`MMC`), and employer-sponsored plans (`ESI`) that some retirees can still access. Our sample includes individuals with prescription drug insurance from these three sources, as well as a control group of Medicare-only (`Medicare`) elderly with no coverage for prescription drugs. The objective is to estimate TES of the three levels of insurance (`inslevel`) that cover prescription drugs compared with Medicare-only (`Medicare`) coverage.

The dependent variable `drugexp` is total annual drug expenditure.

```
. * Outcome - drug expenditure
. qui use mus224mcbs, clear
. generate drugexp = aamttot
. summarize drugexp
```

Variable	Obs	Mean	Std. dev.	Min	Max
drugexp	7,664	1689.965	1890.913	0	54933.82

The distribution is very right skewed, leading to a standard deviation greater than the mean, and 94% of the sample has positive drug expenditures.

The treatment variable `clevel` is the type of insurance. We order this by generosity to ease interpretability of results, though this makes no difference to the empirical results because the methods used in this section treat the insurance categories as unordered.

```
. * Create treatment variable clevel - insurance ordered by increasing generosity
. qui generate clevel= 0 if coverage == "Medicare" // Medicare
. qui replace clevel = 1 if coverage == "MMC"      // MMC (Medicare managed plan)
. qui replace clevel = 2 if coverage == "Medigap"   // Medigap (Medicare suppl ins)
. qui replace clevel = 3 if coverage == "ESI"       // ESI (employer-sponsored)
. tabulate clevel
```

clevel	Freq.	Percent	Cum.
0	1,054	13.75	13.75
1	1,170	15.27	29.02
2	2,354	30.72	59.73
3	3,086	40.27	100.00
Total	7,664	100.00	

We contrast mean expenditure across insurance by running an OLS regression of expenditure on insurance level.

```
. * Variation in mean drug expenditure by type of insurance
. generate dmedicare = clevel == 0
. generate dmmc = clevel == 1
. generate dmedigap = clevel == 2
. generate desi = clevel == 3
. regress drugexp dmmc dmedigap desi, noheader vce(robust)
```

drugexp	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
dmmc	185.7296	55.95032	3.32	0.001	76.05164	295.4075
dmedigap	441.2021	47.57151	9.27	0.000	347.949	534.4553
desi	1283.898	56.569	22.70	0.000	1173.007	1394.789
_cons	1009.119	37.82362	26.68	0.000	934.9748	1083.264

Average drug expenditures for those with only Medicare are \$1,009. Compared with this base category, the average drug expenditures are \$186 higher for Medicare managed plans, \$441 higher for Medigap plans, and \$1,284 higher for employer-sponsored plans.

24.10.3 RA estimates

We use age, gender, ethnicity, income, and health status (measured on a 1 to 5 scale) as regressors.

```
. * Variation in mean drug expenditure by type of insurance
. global xlist h_age h_male h_white income_c genhelth
. summarize $xlist
```

Variable	Obs	Mean	Std. dev.	Min	Max
h_age	7,664	77.22599	7.239356	65	104
h_male	7,664	.4579854	.4982642	0	1
h_white	7,664	.9060543	.2917722	0	1
income_c	7,664	29855.9	45852.22	0	2000000
genhelth	7,664	2.624478	1.060683	1	5

Because the data are so right skewed, we view a better model for the conditional mean of drug expenditures to be an exponential model rather than a linear model. So we use the poisson option of the teffects ra

command rather than the default of OLS. This does not require that the data be Poisson distributed (see section [20.2.4](#)) and has the advantage compared with a log-linear model of including observations with zero drug expenditures.

```
. * RA ATE
. teffects ra (drugexp $xlist, poisson) (clevel), nolog
Treatment-effects estimation                                         Number of obs      =      7,664
Estimator      : regression adjustment
Outcome model  : Poisson
Treatment model: none
```

drugexp	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
ATE						
clevel						
(1 vs 0)	157.6892	56.31752	2.80	0.005	47.30889	268.0695
(2 vs 0)	365.7971	48.27608	7.58	0.000	271.1777	460.4165
(3 vs 0)	1200.963	54.23313	22.14	0.000	1094.668	1307.258
P0mean						
clevel						
0	1065.266	39.99379	26.64	0.000	986.8801	1143.653

The ATE is estimated for each insured group relative to the Medicare (no drug insurance) benchmark. As expected, the ATE is the largest for the ESI category (\$1,201) and the smallest for the MMC category (\$158). The three estimates are statistically significantly different at level 0.05 from the benchmark Medicare only, and their respective 95% confidence intervals do not overlap.

The estimates are similar to those found by the simpler OLS on dummies regression given earlier, with \$1,009 for Medicare-only and, respectively, \$186, \$441, and \$1,284 for the three types of prescription drug coverage.

To generate the ATET (ATE conditional on receiving treatment), we add the `atet` option to the previous command.

```

. * RA ATET
. teffects ra (drugexp $xlist, poisson) (clevel), atet nolog
Treatment-effects estimation                               Number of obs      =      7,664
Estimator       : regression adjustment
Outcome model   : Poisson
Treatment model: none

```

drugexp	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
ATET						
clevel						
(1 vs 0)	181.5681	54.92571	3.31	0.001	73.91567	289.2205
(2 vs 0)	379.1871	47.39422	8.00	0.000	286.2961	472.0781
(3 vs 0)	1187.521	53.12349	22.35	0.000	1083.401	1291.641
POmean						
clevel						
0	1013.281	40.03075	25.31	0.000	934.8222	1091.74

The resulting point and interval estimates are not substantially different from the ATE estimates. A significant difference between ATE and ATET estimates would suggest that self-selection into treatment may be an important factor at work.

To generate the POM for each category, we add the `pomeans` option to the main command.

```

. * RA POMs
. teffects ra (drugexp $xlist, poisson) (clevel), pomeans nolog
Treatment-effects estimation                               Number of obs      =      7,664
Estimator       : regression adjustment
Outcome model   : Poisson
Treatment model: none

```

drugexp	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
POmeans						
clevel						
0	1065.266	39.98537	26.64	0.000	986.8966	1143.636
1	1222.956	40.26935	30.37	0.000	1144.029	1301.882
2	1431.064	28.2025	50.74	0.000	1375.788	1486.339
3	2266.23	38.39453	59.02	0.000	2190.978	2341.482

The pairwise comparisons reported above can also be obtained by generating the contrasts between levels of treatment, using the `contrast` command.

The command `contrast r.clevel` generates a table of ATES relative to the base-level category that can be either the default category or a category specified using the `base()` option.

```
. * Contrasts of TEs after RA compared with base category
. contrast r.clevel, nowald
warning: cannot perform check for estimable functions.

Contrasts of marginal linear predictions
Margins: asbalanced
```

	Contrast	Std. err.	[95% conf. interval]
P0means			
clevel			
(1 vs 0)	157.6892	56.28316	47.37623 268.0022
(2 vs 0)	365.7971	48.27266	271.1844 460.4097
(3 vs 0)	1200.963	54.231	1094.672 1307.254

The alternative command `contrast ar.clevel` compares the ATE for each level of treatment with the adjacent level, which can be useful if the categories are ordered.

```
. * Contrasts of TEs after RA compared with adjacent category
. contrast ar.clevel, nowald
warning: cannot perform check for estimable functions.

Contrasts of marginal linear predictions
Margins: asbalanced
```

	Contrast	Std. err.	[95% conf. interval]
P0means			
clevel			
(1 vs 0)	157.6892	56.28316	47.37623 268.0022
(2 vs 1)	208.1079	48.70919	112.6396 303.5762
(3 vs 2)	835.1662	46.80351	743.433 926.8994

The largest impact comes from moving from `clevel 2` to `3`, a move from Medigap to ESI insurance.

24.10.4 AIPW estimates

The rationale for using the AIPW method in the MLT setting is the same as in the single level case; see section [24.6.5](#). Because the treatment has more than two categories, a multinomial model is needed to predict probabilities for each category for each individual. This requires a probability weight for each sample observation. The `teffects aipw` command uses an MNL specification that is the simplest model for unordered categories.

For illustrative purposes, we treat insurance choice as exogenous, after controlling for insurance premia and the degree of market penetration of Medicare managed care in the individual's region.

Variable name	Storage type	Display format	Value label	Variable label
prem1	double	%12.0g		Prem. of ESI w/ RX
prem2	double	%12.0g		Prem. of ESI w/o RX
prem3	double	%12.0g		Prem. of Medigap w/ RX
prem4	double	%12.0g		Prem. of Medigap w/o RX
penet	double	%12.0g		MMC penetration rate in county from previous year

. summarize \$zlist					
Variable	Obs	Mean	Std. dev.	Min	Max
prem1	7,664	67.01624	46.26629	0	489
prem2	7,664	92.2942	54.32717	0	354
prem3	7,664	170.16	67.30359	0	600
prem4	7,664	133.8445	38.1126	0	570
penet	7,664	11.42253	13.83415	.0039809	49.51

Before using `teffects aipw`, we fit the multinomial model using the `mlogit` command to check the predicted probabilities.

```

. * Predicted probabilities from the MNL model
. qui mlogit clevel $zlist, base(0)
. predict pshat1 pshat2 pshat3 pshat4
(option pr assumed; predicted probabilities)
. summarize dmedicare dmmc dmedigap desi pshat1 pshat2 pshat3 pshat4, sep(4)

```

Variable	Obs	Mean	Std. dev.	Min	Max
dmedicare	7,664	.1375261	.3444244	0	1
dmmc	7,664	.1526618	.3596847	0	1
dmedigap	7,664	.3071503	.4613424	0	1
desi	7,664	.4026618	.4904658	0	1
pshat1	7,664	.1375261	.0409228	.0210833	.2081109
pshat2	7,664	.1526618	.2060478	.0196888	.8223644
pshat3	7,664	.3071503	.0962368	.0461926	.5731233
pshat4	7,664	.4026618	.080105	.1045894	.5314694

The predicted probabilities on average equal the sample frequencies, a property of the MNL model. Some of the predicted probabilities are close to 0, giving some observations great weight, though nowhere near as close to 0 as the default tolerance of 0.00001 used by the `teffects aipw` command.

The `teffects aipw` command yields the following ATE estimates:

```

. * Augmented IPW ATE
. teffects aipw (drugexp $xlist, poisson) (clevel $zlist)
Iteration 0: EE criterion = 3.550e-19
Iteration 1: EE criterion = 2.808e-25
convergence not achieved
The Gauss-Newton stopping criterion has been met but missing standard errors
indicate some of the parameters are not identified.

Treatment-effects estimation                               Number of obs      =    7,664
Estimator        : augmented IPW
Outcome model   : Poisson by ML
Treatment model: (multinomial) logit

```

drugexp	Coefficient	Robust				[95% conf. interval]
		std. err.	z	P> z		
ATE						
clevel						
(1 vs 0)	246.6138	76.06505	3.24	0.001	97.52905	395.6986
(2 vs 0)	370.0061	51.55025	7.18	0.000	268.9694	471.0427
(3 vs 0)	1209.351	58.194	20.78	0.000	1095.293	1323.409
P0mean						
clevel						
0	1050.274	41.47196	25.32	0.000	968.99	1131.557

Warning: Convergence not achieved.

We check covariate balance following use of the inverse-probability weights.

```

. * Check balance following use of the inverse-probability weights
. tebalance summarize
Covariate balance summary

```

Treatment	Observations	
	Raw	Weighted
Obn.clevel =	1,054	2,032.0
1.clevel =	1,170	1,550.2
2.clevel =	2,354	2,057.1
3.clevel =	3,086	2,024.7
Total =	7,664	7,664.0

	Standardized differences		Variance ratio	
	Raw	Weighted	Raw	Weighted
1.clevel				
prem1	.0699667	.0278279	1.319427	1.146251
prem2	-.0242137	-.0696167	1.351879	1.063149
prem3	.2159224	.1015362	1.03429	1.250435
prem4	.2360121	.0104144	1.510058	.8858702
penet	1.871311	.336323	1.228896	.7590858
2.clevel				
prem1	.1023513	-.0015931	1.139751	.978487
prem2	.0879926	.0065958	1.115772	1.093536
prem3	-.0226479	-.0013236	.8979095	.9102739
prem4	.0444557	.0283568	.8151792	.8676917
penet	-.0290098	.0342481	.9459218	1.078047
3.clevel				
prem1	.020812	.0136956	1.068847	.9948594
prem2	.1175974	-.0218133	1.274272	1.197675
prem3	-.0082211	-.0063766	.9894477	.9738561
prem4	-.0010937	.0082907	1.034581	1.043341
penet	.1328474	-.0105397	1.162845	.9723018

The first set of output indicates that the Medicare and MMC plan observations are substantially upweighted and the Medigap and ESI observations are substantially downweighted. The remaining output indicates that the weighted variance ratios differ from one, so a better model for the propensity scores is warranted.

The output from the `teffects aipw` command indicates convergence problems. This problem disappears if we fit the simpler IPW model using command `teffects ipw (drugexp) (clevel $zlist)`. We do not pursue this further.

An extension that controls for potential endogeneity of the level of insurance in this application is given in section [25.3](#).

24.11 Conditional quantile TEs

The chapter has focused on estimating mean TES. Quantile TES were introduced in section 15.5. As noted there, to interpret the quantile TE as measuring the TE for a given individual, one needs to make the strong assumption of rank invariance or rank preservation that is discussed further in section [25.8.4](#).

The community-contributed `poparms` command ([Cattaneo, Drukker, and Holland 2013](#)) calculates conditional quantile TES, under the assumption of conditional independence, using inverse propensity-score weighting or using an efficient-influence-function estimator that includes nonparametric estimation of potential outcomes and is a doubly robust estimator qualitatively similar to AIPW.

We apply this command to the current multilevel example; it also applies to simpler binary treatment examples. The presentation is brief, and we use program defaults; for details, see [Cattaneo, Drukker, and Holland \(2013\)](#).

The following `poparms` command with the `ipw` option gives identical IPW estimates and standard errors of the ATES as the `teffects ipw` command. For brevity, the output is omitted.

```
. * IPW using community-contributed poparms gives same results as teffects ipw
. poparms (clevel $zlist) (drugexp), ipw
. contrast r.clevel, nowald
. teffects ipw (drugexp) (clevel $zlist)
```

Quantile TES estimated by IPW can be obtained using the `quantile()` option. We consider just the median, though simultaneous estimation for several quantiles is possible. The authors find that for the quantile estimates, bootstrap standard errors are more robust than analytical standard errors, though computational time can be considerable. We obtain

```

. * IPW estimates at selected quantiles using community-contributed command poparms
. set seed 10101
. poparms (clevel $zlist) (drugexp $xlist), ipw quantiles(.50)
>     vce(bootstrap, reps(400))

```

Treatment Mean and Quantiles Estimation Number of obs = 7,664
(inverse probability weighting)

drugexp	bootstrap					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
mean						
clevel						
0	983.4526	48.26087	20.38	0.000	888.863	1078.042
1	1273.651	61.46666	20.72	0.000	1153.178	1394.123
2	1439.942	35.29639	40.80	0.000	1370.762	1509.121
3	2284.622	39.7769	57.44	0.000	2206.66	2362.583
q50						
clevel						
0	575.99	48.78364	11.81	0.000	480.3758	671.6042
1	833.01	49.42015	16.86	0.000	736.1483	929.8717
2	1113.86	37.59761	29.63	0.000	1040.17	1187.55
3	1753.56	35.89961	48.85	0.000	1683.198	1823.922

The output provides potential outcomes for the means automatically, in addition to the requested median potential outcomes. As expected, for right-skewed medical expenditure data, the median estimates are less than those for the mean.

To then obtain TES at the median, we apply the `margins`, `pwcompare` command to the results in the second reported equation.

```

. * TEs for the 25th quantile
. margins i.clevel, pwcompare predict(equation(#2))
warning: cannot perform check for estimable functions.

Pairwise comparisons of adjusted predictions           Number of obs = 7,664
Model VCE: bootstrap

Expression: Linear prediction, predict(equation(#2))

```

	Contrast	Delta-method		Unadjusted	
		std. err.	[95% conf. interval]		
clevel					
1 vs 0	257.02	65.22931	129.1729	384.8671	
2 vs 0	537.87	62.16622	416.0264	659.7135	
3 vs 0	1177.57	61.5219	1056.989	1298.151	
2 vs 1	280.85	63.10141	157.1735	404.5265	
3 vs 1	920.55	59.07649	804.7623	1036.338	
3 vs 2	639.7001	51.60011	538.5657	740.8344	

To instead obtain the efficient-influence-function doubly robust estimates, omit the `ipw` option from the `poparm`s command.

24.12 Additional resources

The main command for TES estimation in linear and standard nonlinear models under the conditional independence assumption is `teffects`. Its variant `eteffects` and the `ERM` commands can handle endogenous regressors and tobit-type selection that is relevant when the data are censored. For matching estimators, the community-contributed programs `pscore` and `att` by [Becker and Ichino \(2002\)](#) include some features that are not available with the `teffects` commands. The community-contributed `kmatch` command ([Jann 2017](#)) supports multivariate-distance and PSM, including coarsened exact matching. The community-contributed `poparms` command ([Cattaneo, Drukker, and Holland 2013](#)) provides quantile TES.

[Imbens and Rubin \(2015\)](#) give a particularly comprehensive coverage of the methods presented in this chapter. Treatment evaluation methods are also presented in [Angrist and Pischke \(2009\)](#), [Wooldridge \(2010, chap. 21\)](#), [Cameron and Trivedi \(2005, chap. 25\)](#), [Cunningham \(2021\)](#), and [Glewwe and Todd \(2022\)](#). [Abadie and Cattaneo \(2019\)](#) and [Huber \(2019\)](#) provide excellent surveys. Especially with observational data, one should not blindly use the `teffects` commands. Appropriate diagnostic checks such as those for covariate balance, propensity-score overlap, assessing the unconfoundedness assumption, and robustness of results should be performed.

24.13 Exercises

1. Consider the examples of size and power calculations given in section [24.3.4](#). Example 1 shows the sample sizes required when significance level is fixed at 5% and required power is 0.80. Reproduce the table associated with this example using $\alpha = 0.10$ and power = 0.90. Explain and interpret the results.
2. Suppose that a particular RCT is restricted by financial considerations to treatment and control groups of size 150. The control group's pretreatment average outcome is 16, with standard deviation of 4. Required power is 0.9. What is the minimum-size TE consistent with this design?
3. Consider the generated dataset and example of section [24.4.4](#). Manually implement the RA method of section [24.6.1](#) by two OLS regressions, compute predictions from these two models, and use these predictions to compute the POMS, ATE, and ATET. Verify that you obtain the same estimates as those from the command `teffects ra, y D x`, using appropriate options of that command.
4. Inspect the covariate balance summary output generated by the Stata command `tebalance summarize` given in section [24.9.4](#). The last two lines of the output suggest that the samples may not be balanced with respect to those two regressors. Use Stata's `tebalance density` postestimation diagnostic command to explore whether this is a potential problem before or after the IPW estimator is applied. Repeat the exercise after applying the IPW-RA estimator.
5. The literature suggests that estimates of TE obtained using PSM may not be robust when the distribution of propensity scores in the treated and untreated groups significantly differs. Taking the specification of the model in section [24.9.6](#), repeat the TE calculation given there with the modification that all observations with propensity scores above 0.70 and below 0.10 are excluded. Compare your results with those based on a recommended trimming design that would drop only the observations with propensity scores outside the (0.1, 0.9) range. Which TE estimates do you regard as most plausible?
6. Trimming samples to achieve better balance is recommended as a method of obtaining more robust TE estimates. This recommendation

also applies to `ipw` and `ipwra` methods because computational instability may result from having a significant number of observations with probability weights close to 0 or close to 1. Apply the sample trimming suggestions in the immediately preceding question to the `ipw` estimator used in section [24.9.2](#) and the `ipwra` estimator used in section [24.9.5](#).

7. In this chapter, the focus was on the impact of a binary treatment variable (`lottery`) on out-of-pocket expenditures (`oop`). We will now designate enrollment into Medicaid, defined by the binary variable `medicaid` (which is related to `lottery`), as an alternative treatment variable. Enrollment into Medicaid is, strictly speaking, not exogenous, but we treat it as such for purposes of this exercise by making the assumption of selection on observables only. In the next chapter we cover methods (including the local average TE estimator) that are appropriate when the treatment variable is endogenous. Using the Stata commands from section [24.7](#) and the controls defined through the same macros `$xlist` `$zlist`, estimate the POM of Medicaid enrollees and of nonenrollees. For the RA model, compare the ATE estimate with the ATET estimate. What would you infer from the difference between the two estimates?
8. Again estimate the ATE, but this time use IPW regression with control variables `$zlist` only. Is this estimate significantly different from the previous one obtained without applying IPW? How would you rationalize the difference (or lack of it) with the ATET estimate. What would you infer from the difference between the two estimates?
9. Again estimate the ATE of `medicaid`, but this time use IPW regression and control variables `$zlist` only. Is this estimate significantly different from the previous one obtained without applying IPW? How would you rationalize the difference (or lack of it)?

Chapter 25

Endogenous treatment effects

25.1 Introduction

Chapter [24](#) presented standard methods for estimation of treatment effects (TES) when the treatment can be viewed as exogenous after inclusion of any control variables. These methods rely on the conditional independence assumption of section [24.5](#) that treatment status is independent of potential outcomes after inclusion of control variables, an assumption also referred to as unconfoundedness, ignorability, or selection on observables.

Many observational data applications are for settings where this assumption is not reasonable, in which case treatment status is called endogenous or nonignorable or selected on unobservables. For example, an individual may enter a training program only if it is expected to increase earnings sufficiently to cover the cost of training, but these individual expectations are not observed by the researcher and may not be fully captured by observable characteristics of the individual. Then the methods of chapter [24](#) lead to inconsistent estimates.

This chapter is devoted to methods for endogenous treatment (ET), using several different methods that vary in the strength of model assumptions and the types of data required. Many of the methods, though not all, assume the existence of an instrument that is correlated with treatment but is uncorrelated with potential outcomes after inclusion of regressors.

We begin with a highly parametric approach in which normal distributions are specified for model errors. The TE is the same for all individuals (homogeneous effects) after controlling for regressors, but treatment is endogenous with a random component that is correlated with the random component of the outcome. This approach has been presented in several places in this book, including the linear simultaneous equation model in chapter 7, the selection model in chapter [19](#), and endogeneity in panel models and several nonlinear models. Despite this overlap, we provide here a unified treatment with emphasis on estimation of the effect of treatment. This highly parametric approach has the attraction of being applicable to models with outcome models that are either linear or nonlinear and treatments that are either discrete or continuous. Furthermore, Stata's

extended regression model (ERM) and ET commands implement this parametric approach for many models.

The remainder of the chapter presents various less parametric “quasi-experimental” methods—namely, local average treatment effects (LATEs), the difference-in-differences (DID) estimator, and regression discontinuity (RD) design. The chapter concludes with some treatment extensions to quantile regression. All of these methods allow TES to differ across individuals (heterogeneous effects).

LATE provides a reinterpretation of instrumental variables (IV) and two-stage least-squares (2SLS) given heterogeneous TES. The DID estimator, introduced in section 4.8, overcomes endogeneity in treatment by using richer panel data or repeated cross-sectional data and making the additional assumption of parallel trends so that the conditional independence assumption holds. RD design is applicable to a special type of endogeneity where treatment occurs when an observable continuous measure, one that directly affects the outcome, crosses a threshold. For example, a high enough test score may lead to treatment, yet the outcome of interest is also determined in part by the test score. Then treatment in the neighborhood of the threshold can be viewed as similar to a randomized controlled trial (RCT).

Estimation using these methods to control for both endogeneity and heterogeneity is in many cases straightforward. LATE uses 2SLS, DID uses ordinary least squares (OLS), and RD design uses local polynomial regression or OLS. The difficulty is in understanding the underlying assumptions and the subtleties of the various methods and their extensions. Furthermore, this is an active area of research, the presentation here is introductory, and refinements to these methods are still being established.

The methods of this chapter presume that the inclusion of observable variables is not enough to control for selection into treatment. Chapter 28 presents machine learning methods that can lead to TES estimates that better control for selection into treatment, making the conditional independence assumption of chapter 24 more tenable.

25.2 Parametric methods for endogenous treatment

If treatment assignment is not ignorable, is the potential-outcome framework still appropriate? Is it meaningful to base causal inference on the framework of counterfactual causality? Extension of the causal parameter concept to nonrandomized observational settings is due to [Rubin \(1974, 1978\)](#). Endogeneity is a significant complication in estimation; however, once it has been handled using a suitable instrument, the treatment parameter is identifiable. Then, it is argued, one may continue to generate the hypothetical potential outcomes based on consistent estimation of the model parameters. Thus, the potential-outcome mean (POM) framework remains relevant even when treatment is endogenously chosen.

Many commonly used models with ET consist of two equations, a linear outcome equation with treatment as an argument and a linear or nonlinear reduced-form treatment participation (or assignment) equation for the treatment variable that includes one or more instruments. Examples of this model structure appeared in chapters 7 and [19](#). If the random components of the two equations are correlated, then the treatment is said to be endogenous. IV estimation for the linear model was extensively dealt with in chapter 7.

In this section, we focus on an empirically important variant of IV in which the ET variable D is discrete, rather than continuous, and the reduced form for D may be nonlinear.

25.2.1 Canonical model

We specify the outcome y_i for the i th subject to be a linear function of observable variables \mathbf{x}_i and an endogenous binary treatment participation decision indicator D_i , with

$$y_i = \alpha D_i + \mathbf{x}'_i \boldsymbol{\beta} + u_i \quad (25.1)$$

Participation depends on the instrumental variable z_i (which may be binary),

$$D_i^* = \gamma_1 z_i + \mathbf{x}'_i \boldsymbol{\gamma}_2 + v_i \quad (25.2)$$

D_i^* is a latent variable that has observable counterpart D_i generated by the observability condition

$$D_i = \begin{cases} 0 & \text{if } D_i^* \leq 0 \\ 1 & \text{if } D_i^* > 0 \end{cases} \quad (25.3)$$

Richer models, also considered below, allow for nonlinear outcome models, interactions between treatment and the regressors in (25.1), an endogenous multivalued treatment, and more than one instrument.

25.2.2 Assumptions

We make the following assumptions in sections [25.2–25.4](#).

1. The instrument z appears in the D -equation and does not appear in the y -equation. This is referred to as an exclusion restriction. The instrument may be continuous or discrete and in a special case is binary.
2. Conditional on (\mathbf{x}, z) , $\text{Cov}(z, v) = \text{Cov}(u, z) = \text{Cov}(\mathbf{x}, u) = 0$, but $\text{Cov}(D, u) \neq 0$ because u and v are correlated. Furthermore, treatment D depends upon the instrument z in a nontrivial fashion; to emphasize dependence of D on z , we use the notation $D(z)$.
3. Responses are homogeneous across subjects; that is, there is no randomness of the coefficient α .

These assumptions can be summarized by stating that the model has a recursive or triangular structure that does not permit simultaneous dependence between y and D . D impacts y , but there is no reverse

feedback to D . Thus, causation runs from the instrument to the treatment and then to outcome. However, the causal connection from the instrument to the treatment has no role in the interpretation of TES. Whereas some models would regard the instruments as the ultimate causal variable, and econometricians may carefully study the mechanism that connects it to the treatment variable, this connection is essentially ignored in the POM framework.

We assume a linear probability model for D or that the distribution of the random error v_i is standard normal, which implies a probit model for the binary treatment D . Under these assumptions, (α, β) are identified, and α estimates both the average treatment effect (ATE) and the average treatment effect on the treated (ATET). These methods that control for $\text{Cov}(D, u) \neq 0$ yield consistent estimates, whereas OLS estimates of (25.1) are biased.

The key features of this model are maintained if the linear outcome (25.1) is replaced by a nonlinear equation such as a probit model for a binary outcome or by an ordered probit model for an ordered discrete outcome. In those cases, the errors in the extension to (25.1) and in (25.3) are assumed to be normally distributed.

The outcome (25.1) can be generalized to a potential-outcomes or regression-adjustment model (see section 24.6.1) that interacts the ET D with all regressors. Then,

$$\begin{aligned} y_{1i} &= \mathbf{x}'_i \boldsymbol{\beta}_1 + u_{1i} && \text{if } D_i = 1 \\ y_{0i} &= \mathbf{x}'_i \boldsymbol{\beta}_0 + u_{0i} && \text{if } D_i = 0 \end{aligned} \tag{25.4}$$

This specifies a restricted form of heterogeneous TES.

In extension of (25.3) to a multivalued treatment model, we check for block triangularity, the treatment equations being regarded as a single block. If the outcome model (25.1) is linear and there are additional nontreatment endogenous regressors, then it may be possible in some cases to achieve triangularity either by reordering of equations or by substituting out some endogenous regressors using their reduced form.

25.2.3 Estimation methods

Given the assumption of joint normality of model errors, the models can be fit by (full-information) maximum likelihood (ML). For more complex models, this can require numerical integration methods.

In some cases, assumptions can be relaxed, typically by assuming normality of individual errors but not necessarily requiring joint normality. Given a probit model for binary treatment, a two-step method obtains an estimate of the inverse Mills ratio that is added as an auxiliary regressor in the outcome equation, similar to the two-step procedure used in selection models; see section [19.6.4](#). Alternatively, a control function approach calculates a residual from the fitted probit model and includes this as an additional regressor in the outcome equation.

25.2.4 Computing TEs

In simple outcome models such as ([25.1](#)), the TE equals the estimated coefficient of the treatment variable.

In more complex models that model separately each potential outcome or specify a nonlinear outcome model, the ATE or ATET can be computed by applying the `margins` command after estimation of the endogenous regressor model; see sections 13.7 and [17.6](#). This was illustrated in the context of binary outcome models (`ivprobit`), selection models (`heckman`), and event count models (nonlinear IV or `gmm`) in their respective chapters; see sections [17.6](#), [19.6](#), and [20.7](#). And the `margins` command provides additional flexibility in defining different counterfactuals that go beyond the standard ATE calculation; see section 13.7.5 for an example.

For some nonlinear models, the `margins` command may not be available. Provided, however, that `predict` or `predictnl` commands, or both, can be used postestimation, the TEs can be estimated by processing output generated by the `predict` command; see section [25.4.5](#) for an example. Because the `predict` command can be conditioned on specific combinations of variables for specific subpopulations if so desired, it is a

particularly convenient and flexible method for generating ATES as well as distributing of TES in both linear and nonlinear models.

The next two sections present Stata ERM and ET commands that fit various models with ET and conveniently compute the subsequent ATE and ATET.

25.3 ERM commands for endogenous treatment

Estimates of TES in models with endogenous (or exogenous) treatments can be obtained using Stata's ERM family of commands described in section [23.7](#). These commands are applicable to the linear model for outcomes, using the `eregress` command, and to the probit, ordered probit and interval regression models that are based on a latent variable model that is linear with a normally distributed error.

25.3.1 The ERM commands

The ERM commands are based on the assumption that the joint distribution of errors in outcome and treatment equations is a multivariate normal distribution. A unique feature of the multivariate normal distribution is that it can be expressed as a product of the conditional normal (for the outcome) and marginal normal (for the treatment). Adding triangularity or recursivity as an additional feature facilitates efficient computation of the model. Estimation is by ML.

An example of a model that can be handled using `eregress` is a continuous outcome model with a binary treatment. To respect bivariate normality, we use the probit specification of the treatment equation and a linear specification of the outcome equation. By contrast, a logit specification of the treatment equation will not work, because the latent variable formulation of a logit model has a logistic distributed error.

We consider the application of the suite of ERM commands for regression-based treatment evaluation in the presence of the following additional complications that may occur individually or jointly: outcome y in [\(25.1\)](#) depends on other endogenous variable (y_2) that may be binary or continuous; treatment (t_1) is endogenous; treatment may be binary or multilevel. Table [25.1](#) presents a flexible range of models and options, including, for example, the presence of ET as well as other endogenous variables. In all examples, it is assumed that errors are jointly normal.

Table 25.1. Selected ERM commands when treatment is endogenous

Examples of Stata-extended commands and optional subcommands

Linear regression with a continuous ET

```
eregress y x, endogenous(t1 = z x)
```

Linear regression with a continuous endogenous regressor and ET

```
eregress y1 x, endogenous(y2 = x z1) endogenous(t1 = z3 x)
```

Linear regression with a continuous endogenous regressor and a binary ET

```
eregress y1 x, endogenous(y2 = x z1) entreat(t1 = z3 x)
```

Linear regression with a continuous endogenous regressor and a multivalued treatment

```
eregress y1 x, endogenous(y2 = x z1) entreat(t2 = z3 x)
```

Linear regression with a continuous endogenous regressor, multivalued treatment, and selection

```
eregress y1 x, endogenous(y2 = x z1) entreat(t2 = z3 x) select(s1=w x)
```

The first two examples in table 25.1 are a linear model with continuous endogenous treatment variable t_1 and instrument z that is excluded from the outcome equation for y . The second example adds a continuous endogenous regressor y_2 to the first model.

The next three examples again have a continuous outcome, but now the ET is discrete. In the third example, the treatment is binary and by default is modeled by a probit model given the assumption of normal errors. In the fourth example, the treatment is multilevel and by default is modeled by an ordered probit model given the assumption of normal errors. An example is three levels of health insurance calibrated in terms of the generosity of coverage.

The `eprobit` and `eoprobit` commands for binary and ordered discrete outcomes with endogenous regressors or ET effects, or both, have syntax that is essentially the same as for extended linear regressions. The extended interval regression (`eintreg`) command is used for tobit-type selection models.

The `estat teffects` postestimation command provides the ATE estimate as default; options provide the estimated ATET and the POMS. If the model is fit with option `vce(robust)`, rather than with default standard errors, then

`estat teffects` gives unconditional standard errors that additionally allow for variation in the regressors; see section 13.7.9.

25.3.2 Interpretation of ET effects

The impact of a change in an endogenous variable, including a treatment variable, is subtly different from that of an exogenous variable. For an exogenous variable, the ATE is defined as the average change in the outcome due to a unit change in the regressor, holding all other regressors constant. In the linear model (25.1), the coefficient of the regressor is the ATE.

When the regressor is endogenous, however, it includes the effect of the individual specific random-error term and, in that sense, is the total effect of the regressor and the effect of the idiosyncratic error v_i in (25.2). Averaging over all impacted observations requires also averaging over the random component. Such averaging is interpreted as application of the average structural function; for details, see [Blundell and Powell \(2003\)](#) and [Wooldridge \(2010, 24–25\)](#). The resulting impact parameter is then called either the average structural mean or, in a binary outcome model, the average structural probability. In a linear outcome model with zero-mean errors, such averaging does not change the usual interpretation of the coefficient. In a nonlinear model, such as a probit outcome model, the average structural function is nonlinear, and averaging involves numerical integration. Thus, even though the model specifies the TE to be the same for all individuals if treatment was exogenous, the endogeneity leads to heterogeneous effects in the case of a nonlinear outcome model.

25.3.3 Endogenous multivalued TEs application

We apply the `eregress` command to the prescription drugs expenditure with the multilevel treatment example previously analyzed in section [24.10](#), where now the multilevel treatment is allowed to be endogenous. ML is used to fit the joint model of outcome and treatment. This involves placing additional structure on the specified model.

Because joint normality is required, the outcome variable is log drug expenditures (`ldrugexp`), and a log-linear outcome model is specified, rather

than analyzing the level of drug expenditures by Poisson regression as in section [24.10](#). The treatment variable is level of health insurance (`clevel`), which takes three discrete values (in addition to basic Medicare) that are now treated as endogenous. An ordered probit model is used to model the choice of health insurance where endogeneity of insurance choice is modeled by allowing for error correlation and by instruments that affect insurance choice but not drug expenditures.

The objective of the analysis is to estimate the ATE and ATET for each type of treatment. The assumption of recursiveness between treatment and outcomes is maintained but also extended by allowing for presence of unobserved correlated random factors that affect both the choice of the treatment `clevel` and outcome `ldrugexp`.

The base level of health insurance is Medicare that only offers zero prescription drug coverage and serves as the baseline counterfactual. The other levels are `MMC` (Medicare managed care), `Medigap` (Medicare supplemental insurance), and `ESI` (employer-sponsored insurance).

The following code drops 7% of the sample with 0 annual drug expenditure and creates the insurance variable (`clevel`) that is ordered by increasing levels of insurance cover.

```
. * Read in data, drop zero drug spending, create ordered multivalued treatment
. qui use mus224mcbs
. generate drugexp = aamttot
. drop if drugexp == 0
(508 observations deleted)
. qui generate ldrugexp = ln(drugexp)
. qui replace income_c = income_c/1000
. qui tabulate coverage, generate(inslevel)
. qui summarize inslevel*
. qui generate clevel= 0
. qui replace clevel=0 if inslevel3==1 // Base Medicare
. qui replace clevel=1 if inslevel2==1 // MMC (Medicare managed plan)
. qui replace clevel=2 if inslevel4==1 // Medigap (Medicare suppl ins)
. qui replace clevel=3 if inslevel1==1 // ESI (Employer sponsored)
```

We then verify that indeed the outcome increases on average with the level of treatment, very substantially for levels 2 and 3 compared with levels 0 and 1. The subsequent analysis will control for individual characteristics and possible endogeneity of insurance choice.

```
. * eregress example: log drug expenditure by insurance type
. regress ldrugexp ibn.clevel, noheader noconstant
```

ldrugexp	Coefficient	Std. err.	t	P> t	[95% conf. interval]
clevel					
0	6.465467	.0402077	160.80	0.000	6.386648 6.544286
1	6.561296	.0359018	182.76	0.000	6.490918 6.631674
2	6.890744	.0253979	271.31	0.000	6.840956 6.940531
3	7.306901	.0219283	333.22	0.000	7.263915 7.349887

There are two sets of regressors. The first set (`xlist`) consists of individual socioeconomic characteristics that are included in both the outcome equation and the treatment choice equation. The second set (`zlist`) consists of regressors (instruments) that are excluded from the outcome equation. These are insurance premiums (`prem1–prem4`) for available plans and a variable (`penet`) reflecting the competition (market penetration) in the local insurance market in which the plan is purchased.

```
. * eregress: Summary stats for outcome, exogenous variables, and instruments
. global xlist h_age h_male income_c genhelth // Exogenous in both
. global zlist prem1 prem2 prem3 prem4 penet // Instruments
. summarize ldrugexp $xlist $zlist
```

Variable	Obs	Mean	Std. dev.	Min	Max
ldrugexp	7,156	6.959769	1.236787	0	10.91388
h_age	7,156	77.30604	7.237295	65	104
h_male	7,156	.4491336	.4974406	0	1
income_c	7,156	30.19316	47.08639	0	2000
genhelth	7,156	2.665735	1.053193	1	5
prem1	7,156	67.07746	46.14253	0	489
prem2	7,156	92.14425	54.41507	0	354
prem3	7,156	169.9314	67.17174	0	600
prem4	7,156	133.9728	37.32712	0	570
penet	7,156	11.43595	13.85367	.0039809	49.51

The `eregress` command with option `entreat(clevel)` recognizes that `clevel` takes multiple discrete values, fits a separate potential-outcome

model for each level of treatment, and uses an ordered probit model for the treatments, with treatment choice endogenous.

```
. * eregress: Endogenous multivalued treatment and regression-adjustment model
. eregress ldrugexp $xlist, entreat(clevel = $xlist $zlist) vce(robust) nolog
Extended linear regression                                         Number of obs =      7,156
                                                               Wald chi2(20) = 270515.14
Log pseudolikelihood = -20024.711                                         Prob > chi2 =      0.0000
```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
ldrugexp						
clevel#c.h_age						
0	.0042307	.0057991	0.73	0.466	-.0071353	.0155968
1	.0035396	.0053761	0.66	0.510	-.0069975	.0140766
2	.0048934	.0031718	1.54	0.123	-.0013231	.01111
3	.0006181	.0032107	0.19	0.847	-.0056747	.0069109
clevel#c.h_male						
0	-.1954782	.0912511	-2.14	0.032	-.3743271	-.0166293
1	-.160039	.0871775	-1.84	0.066	-.3309038	.0108257
2	-.077942	.0575325	-1.35	0.175	-.1907035	.0348196
3	-.1556137	.0428371	-3.63	0.000	-.2395729	-.0716545
clevel#						
c.income_c						
0	.0022249	.0017465	1.27	0.203	-.0011981	.0056479
1	-.0003385	.0021949	-0.15	0.877	-.0046405	.0039635
2	.000125	.0018354	0.07	0.946	-.0034723	.0037222
3	-.000383	.0006372	-0.60	0.548	-.001632	.0008659

clevel#						
c.genhelth						
0	.336533	.0383245	8.78	0.000	.2614183	.4116477
1	.3608828	.03661	9.86	0.000	.2891284	.4326371
2	.2831774	.0228076	12.42	0.000	.2384752	.3278795
3	.2948639	.0201427	14.64	0.000	.255385	.3343428
clevel						
0	4.80542	.7802745	6.16	0.000	3.276111	6.33473
1	5.252532	.4904076	10.71	0.000	4.29135	6.213713
2	5.717317	.2771843	20.63	0.000	5.174046	6.260588
3	6.812011	.3253975	20.93	0.000	6.174244	7.449779
clevel						
h_age	-.0013482	.0018959	-0.71	0.477	-.0050641	.0023677
h_male	-.0998152	.0287945	-3.47	0.001	-.1562514	-.0433789
income_c	.0054154	.0015643	3.46	0.001	.0023494	.0084814
genhelth	-.0078822	.0132423	-0.60	0.552	-.0338366	.0180723
prem1	.0001451	.0002809	0.52	0.605	-.0004055	.0006957
prem2	.0009054	.0002899	3.12	0.002	.0003371	.0014737
prem3	-.000038	.0002012	-0.19	0.850	-.0004324	.0003565
prem4	-.0006851	.0004436	-1.54	0.123	-.0015546	.0001844
penet	-.0137505	.0011206	-12.27	0.000	-.0159469	-.0115541
/clevel						
cut1	-1.387702	.1778247			-1.736232	-1.039172
cut2	-.7940095	.1783897			-1.143647	-.4443721
cut3	.0619699	.1789969			-.2888576	.4127974
var(e.ldrugexp)	1.372352	.1577232			1.095564	1.71907
corr(e.clevel, e.ldrugexp)	-.2350591	.2832739	-0.83	0.407	-.6789779	.3347248

The first set of output gives the fitted coefficients for the outcome model at each level of insurance. For example, the coefficient of `age` in the outcome equation fitted for individuals with `clevel = 2` is 0.00489, so for those individuals, drug expenditures increase by approximately 0.5% with each year of aging. The second set of output gives the ordered probit fitted regression coefficients and the three cutoffs.

The last line shows that the covariance between the two equation errors is negative and is statistically insignificant at the 5% level. Negative correlation means that unobserved random factors that increase the probability of higher level of insurance actually decrease the probability of higher expenditure. This is not consistent with adverse selection. It is

consistent with, for example, unobserved risk aversion that leads to both choice of more generous health insurance and healthier habits.

Next we use the `estat teffects` command to estimate ATE for the three insurance levels relative to no insurance beyond basic Medicare.

```
. * egress: ATE for endogenous multivalued treatment with unconditional
> standard errors
. estat teffects, ate
```

Predictive margins		Number of obs = 7,156				
		Unconditional				
		Margin	std. err.	z	P> z	[95% conf. interval]
ATE	clevel					
	(1 vs 0)	.397106	.3890177	1.02	0.307	-.3653546 1.159567
	(2 vs 0)	.8102799	.5270496	1.54	0.124	-.2227183 1.843278
	(3 vs 0)	1.555395	.9227738	1.69	0.092	-.2532084 3.363998

For MMC, Medigap and ESI the ATE is estimated to be, respectively, 39.7%, 81.0%, and 155.5% higher than for those in Medicare only. The results for ATET, obtained using option `atet`, are within one percent of the ATE estimates.

The preceding estimates are based on model estimation using robust standard errors. If instead we use default standard errors in estimation we obtain the same ATE estimates but the standard errors are considerably smaller.

```

. * eregress: ATE following regression with default standard errors had
> smaller standard errors
. qui eregress ldrugexp $xlist, entreat(clevel = $xlist $zlist) nolog
. estat teffects, ate
Predictive margins                                         Number of obs = 7,156
Model VCE: OIM

```

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
ATE						
clevel						
(1 vs 0)	.397106	.1897812	2.09	0.036	.0251416	.7690704
(2 vs 0)	.8102799	.254762	3.18	0.001	.3109556	1.309604
(3 vs 0)	1.555395	.445699	3.49	0.000	.681841	2.428949

Note: Standard errors treat sample covariate values as fixed and not a draw from the population. If your interest is in population rather than sample effects, refit your model using `vce(robust)`.

The output from `estat teffects` includes a statement that the standard errors are specific to the sample, which is the usual approach in applied microeconomics studies. For this example, however, the larger standard errors in the initially provided ATE estimates are due to the robust standard errors for the error variance and covariance being larger, rather than due to allowing covariate values being a draw from the population.

The preceding potential-outcomes model specifies that the mean outcome for those with insurance level j is $y_i = \alpha_j + \mathbf{x}'_i \beta_j, j = 0, 1, 2, 3$. A less flexible model that restricts $\beta_j = \beta$ and specifies that $y_i = \sum_{j=1}^3 \alpha_j D_{ij} + \mathbf{x}'_i \beta$ can be estimated using the `nointeract` option of `entreat()`. In that case, the fitted coefficients $\hat{\alpha}_j$ and their standard errors are equivalent to the estimated ATE and ATET and their standard errors. We obtain

```

. * eregress: Endogenous multivalued treatment & simpler no interaction model
. qui eregress ldrugexp $xlist, entreat(clevel = $xlist $zlist, nointeract)
>      vce(robust)
. estat teffects, ate
Predictive margins                                         Number of obs = 7,156

```

	Unconditional					
	Margin	std. err.	z	P> z	[95% conf. interval]	
ATE						
clevel						
(1 vs 0)	.5264812	.3014017	1.75	0.081	-.0642552	1.117218
(2 vs 0)	.9794175	.4106409	2.39	0.017	.1745761	1.784259
(3 vs 0)	1.833951	.7217249	2.54	0.011	.4193963	3.248506

The TES are 10%–20% larger than in the potential-outcomes variant of this model. The more restricted model leads to greater precision because the standard errors are approximately 25% smaller.

The reader is reminded that these results are obtained using a dataset that excludes 508 sample individuals who reported zero expenditure, which potentially creates a tobit-type selection problem; this can be tackled by adding the `select()` option to the `eregress` command.

The `eregress` command is flexible. It can simultaneously allow for endogenous (nontreatment) regressors, ET, or exogenous treatments. The key restrictions are the recursive structure and normal distribution assumption. If a specified model is not recursive, it can be made recursive (or triangular) using some “tricks” of triangulation; see the [ERM] *Stata Extended Regression Models Reference Manual*, “How to triangularize a system of equations.”

25.4 ET commands for binary endogenous treatment

An alternative to the ERM commands are the ET commands. These commands are restricted to binary treatment modeled by a probit model. The three commands are `etregress` for a linear outcome model, `etpoisson` for an exponential conditional mean outcome model, and `eteffects` for a range of outcome models summarized below.

25.4.1 The `etregress` command

The `etregress` command has the following syntax:

```
etregress depvar [indepvars], treat(depvar_t = indepvars_t) [, options]
```

where the subscript t denotes the treatment equation variables.

A simple application is `etregress y x, treat(D = x z)`, which applies to the canonical model in section [25.2.1](#). The `poutcomes` option allows the outcome model errors to vary with treatment status [see [\(25.4\)](#)], specifying a trivariate normal distribution for the errors u_{0i} , u_{1i} , and v_i . To allow interaction between the treatment variable and the outcome regressor, one needs to explicitly define the interactions, for example, `etregress y i.D#(c.x1 c.x2), treat(D = c.x1 c.x2 z)`.

Three estimation methods are available.

The default option is ML estimation based on joint normality of the errors u_i and v_i in [\(25.1\)](#) and [\(25.2\)](#). This yields exactly the same result as the `erregress` command.

The option `twostep` delivers estimates based on a two-step estimator in which a probit model is fit first, an estimate of the inverse Mills ratio (“lambda”) or hazard of receiving treatment is fit for each observation, and at the second stage, the outcome equation is reestimated with “lambda” added as an auxiliary variable. This two-step estimator relaxes the joint normality assumption of the errors and parallels the so-called heckit procedure used in selection models; see section [19.6.4](#).

The option `cfunction` yields the same estimates as option `twostep` but does so by stacking all equations and fitting the model by (just-identified) generalized methods of moments, similar to the linear control function example in section 13.3.11.

Note that if the treatment equation is expressed as a linear probability model, rather than a probit model, then Stata's `ivregress 2sls` command yields an alternative 2SLS estimate of the TE. The `margins` postestimation command yields various TES estimates following estimation of the model. For the `margins` command, option `predict(cte)` gives the ATE, and the additional option `subpop()` enables estimation of the ATET. The option `predict(yctr)` gives $E(y|\text{treated})$, option `predict(ycntr)` gives $E(y|\text{untreated})$, and option `predict(ptrt)` gives the average probability of treatment. If model estimates are obtained using option `vce(robust)`, rather than with default standard errors, then `margins` gives unconditional standard errors that additionally allow for variation in the regressors; see section 13.7.9.

25.4.2 Endogenous binary treatment application

The dataset is a 2003 extract from the Medical Expenditure Panel Survey of individuals over the age of 65 years analyzed in section 7.4.2. The outcome equation has the dependent variable `ldrugexp`, the log of total out-of-pocket expenditures on prescribed medications. The binary ET variable is an indicator for whether the individual holds either employer- or union-sponsored health insurance (`hi_empunion`). Other regressors are number of chronic conditions (`totchr`) and four sociodemographic variables: age in years (`age`), dummy variables for female (`female`) and black or Hispanic (`blhisp`), and the natural logarithm of annual household income in thousands of dollars (`linc`). As an instrument for insurance (`hi_empunion`), we use a measure of affordability—the ratio of social security income to total income (`ssiratio`).

Supplementary insurance for drug coverage is a choice variable for the near universal Medicare insurance for the elderly. Conditional correlation between `hi_empunion` and `ldrugexp` could arise if those who expect high

future drug expenses are more likely to choose a job that provides supplementary health insurance upon retirement.

The model fit is identical to that in section [25.2.1](#), with treatment appearing as a single regressor, rather than as a regression-adjustment model with interaction with all regressors. The ML estimates, along with the first-stage probit estimates that indicate that the chosen instrument is robust, are as follows.

```
. * etregress: ML for endogenous binary treatment (insurance)
. qui use mus207mepspresdrugs, clear
. global x2list totchr age female blhispl linc
. etregress ldrugexp $x2list, treat(hi_empunion = $x2list ssiratio)
>     vce(robust) first nolog
```

Probit regression

	Number of obs = 10,068
	LR chi2(6) = 829.98
	Prob > chi2 = 0.0000
	Pseudo R2 = 0.0620

Log likelihood = -6282.5099

hi_empunion	Coefficient	Std. err.	z	P> z	[95% conf. interval]
totchr	.0370418	.0101307	3.66	0.000	.0171861 .0568976
age	-.0240092	.0020265	-11.85	0.000	-.027981 -.0200374
female	-.2015789	.0264087	-7.63	0.000	-.253339 -.1498189
blhispl	-.1847738	.0367088	-5.03	0.000	-.2567217 -.1128259
linc	.1216865	.0157349	7.73	0.000	.0908468 .1525263
ssiratio	-.6169911	.041986	-14.70	0.000	-.699282 -.5347001
_cons	1.552571	.1621934	9.57	0.000	1.234678 1.870464

Linear regression with endogenous treatment
Estimator: Maximum likelihood
Log pseudolikelihood = -22654.396

Number of obs = 10,068
Wald chi2(6) = 1876.40
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	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
	ssiratio	-.5506714	.0400378	-13.75	0.000	-.629144 -.4721987
	_cons	1.508993	.1649894	9.15	0.000	1.18562 1.832367
/athrho	/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 estimated correlation `rho` between equation errors is 0.643 and is very precisely estimated; the Wald chi-squared statistic for test of zero error correlation is 155.51. The output includes `sigma`, the estimated standard deviation of the output equation error, and a quantity `lambda`, which is simply `rho times sigma`.

Because the outcome equation is linear, the ATE and ATET are identical and equal the estimated coefficient of – 1.384, the coefficient of `hi_empunion`. For illustrative purposes, we nonetheless use the `margins` postestimation command to obtain the average marginal effect (AME).

```

. * etregress: ATE of insurance status following ML estimation
. margins, predict(cte)
Predictive margins                                         Number of obs = 10,068
Model VCE: Robust
Expression: Conditional treatment effect, predict(cte)


```

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
_cons	-1.384199	.1029986	-13.44	0.000	-1.586072	-1.182325

If the model specification included interaction terms involving the endogenous dummy variable and other regressors, as in (25.4), then both ATE and ATET would be different from the fitted slope coefficient.

The `cfunction` option yields the following estimates of the ATE and ATET.

```

. * etregress: Control function estimator for endog binary treatment (insurance)
. qui etregress ldrugexp $x2list, treat(hi_empunion = $x2list ssiratio)
>      cfunction vce(robust)
. margins, predict(cte)
Predictive margins                                         Number of obs = 10,068
Model VCE: Robust
Expression: Conditional treatment effect, predict(cte)


```

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
_cons	-.8711352	.1824591	-4.77	0.000	-1.228748	-.5135219

The estimated TE is -0.871 , substantially lower than -1.384 estimated by ML. The estimated confidence interval is also wider. The estimator uses an estimated value of the inverse Mills ratio or hazard function from the probit regression as an additional variable to control for endogeneity. From output not included, the coefficient is highly significant, again confirming endogeneity of the insurance variable.

The `twostep` option leads to the same parameter estimates as the `cfunction` option; the `twostep` option provides only default standard errors, while `cfunction` can additionally yield heteroskedastic-robust standard errors.

The following example fits a richer model that interacts treatment status with control variables and uses option `poutcomes` to allow the outcome model errors to vary with treatment status. The subsequent `margins` command includes the `subpop()` option to obtain the ATET.

```
. * etregress: More flexible potential-outcomes model and ATET
. global x3list c.totchr c.age i.female i.blhisp c.linc
. qui etregress ldrugexp i.hi_empunion#($x3list),
>     treat(hi_empunion = $x3list ssiratio) vce(robust) nolog poutcomes
. margins, predict(cte) subpop(if hi_empunion==1)
Predictive margins                                         Number of obs     = 10,068
Model VCE: Robust                                         Subpop. no. obs =  3,850
Expression: Conditional treatment effect, predict(cte)


```

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
_cons	-2.453818	.0502733	-48.81	0.000	-2.552352	-2.355284

The ATET equals -2.454 .

25.4.3 The `etpoisson` command

The `etpoisson` command has essentially the same syntax as the `etregress` command.

The model is the same as the canonical model (25.1)–(25.3) in section 25.2.1, except that (25.1) is replaced by

$$y_i = \exp(\alpha D_i + \mathbf{x}'_i \boldsymbol{\beta} + u_i) \quad (25.5)$$

The error terms u_i in (25.5) and v_i in (25.2) are assumed to be jointly normal distributed with correlation ρ . The model can be fit by ML. Using properties of the normal distribution, one can find analytical expressions for the ATE and ATET from this model.

While the command is named `etpoisson`, it is more precisely an `et` command for outcome with exponential conditional mean. If y is actually a

count, the model is still suitable because it has the same exponential conditional mean as a Poisson regression model and controls for overdispersion because $E\{\exp(u_i)\} > 1$ if u_i is normal with mean zero, leading to $\text{Var}(y_i|D_i, \mathbf{x}_i) > E(y_i|D_i, \mathbf{x}_i)$.

25.4.4 Application to count data with ET

We apply the `etpoisson` command to the number of doctor visits data from the U.S. Medical Expenditure Panel Survey 2003, which was analyzed in chapter 10.

The Poisson regression has dependent variable `docvis` and endogenous regressor `private` (insurance) with exogenous regressors `female`, `chronic` (number of chronic conditions), and `age` (in years divided by 10).

For this illustration, we work with a random sample of 2,000 observations and truncate the number of doctor visits at 50. The data summary follows.

```
. * etpoisson sample: Subsample of young MEPS sample
. qui use mus210mepsdocvisyoung, clear
. set seed 10101
. qui sample 2000, count
. keep if docvis < 50
(7 observations deleted)
. regress docvis private, vce(robust) noheader
```

docvis	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
private	2.59169	.2744089	9.44	0.000	2.053531	3.129849
_cons	1.415042	.2222057	6.37	0.000	.9792618	1.850822

The average number of doctor visits is 1.415 for those without private insurance and is 2.592 higher for those with private insurance. Without control variables, the AME or ATE is the slope estimate 2.592.

We first obtain Poisson regression estimates under the assumption that `private` is exogenous; only the marginal effect (ME) is reported.

```

. * poisson: AME = ATE of exogenous binary treatment using margins and finite
> diffs
. qui poisson docvis i.private i.chronic i.female age, vce(robust)
. margins, dydx(i.private)
Average marginal effects                                         Number of obs = 1,993
Model VCE: Robust
Expression: Predicted number of events, predict()
dy/dx wrt: 1.private

```

	Delta-method					
	dy/dx	std. err.	z	P> z	[95% conf. interval]	
1.private	2.316279	.2860775	8.10	0.000	1.755577	2.87698

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

The AME is 2.316 following Poisson regression with control variables included and is highly statistically significant.

Next we assume endogeneity of `private` and specify as instrument `income` (annual income in thousands of dollars). This example is purely illustrative because in practice `income` can be expected to have a direct effect on the number of doctor visits.

```

. * etpoisson: ML for endogenous binary treatment
. etpoisson docvis i.chronic i.female age,
>     treat(private= i.chronic i.female age income) vce(robust) nolog
Poisson regression with endogenous treatment          Number of obs = 1,993
(24 quadrature points)                               Wald chi2(4) = 1066.08
Log pseudolikelihood = -5085.0789                  Prob > chi2 = 0.0000

```

	Robust					
	Coefficient	std. err.	z	P> z	[95% conf. interval]	
docvis						
1.chronic	1.360064	.1121328	12.13	0.000	1.140287	1.57984
1.female	.6409438	.0750489	8.54	0.000	.4938508	.7880369
age	.1144535	.0526475	2.17	0.030	.0112663	.2176407
1.private	1.431485	.131891	10.85	0.000	1.172983	1.689986
_cons	-2.086742	.1580321	-13.20	0.000	-2.39648	-1.777005
private						
1.chronic	.229521	.0794046	2.89	0.004	.0738909	.3851511
1.female	.3199315	.0729008	4.39	0.000	.1770486	.4628143
age	.0795119	.0356403	2.23	0.026	.0096581	.1493657
income	.0319134	.0044206	7.22	0.000	.0232492	.0405775
_cons	-.4277604	.1745324	-2.45	0.014	-.7698376	-.0856833
/athrho	-.1764639	.0700389	-2.52	0.012	-.3137376	-.0391903
/lnsigma	.1740968	.0200664	8.68	0.000	.1347673	.2134264
rho	-.1746548	.0679024			-.3038335	-.0391702
sigma	1.190171	.0238825			1.144271	1.237912

Wald test of indep. eqns. (rho = 0): chi2(1) = 6.35 Prob > chi2 = 0.0118

The correlation coefficient (`rho` in the table) for the errors is -0.17 and is statistically significant at 5%, confirming strong evidence that `private` should be treated as an endogenous variable.

We can compute the associated ATE using the `margins` command.

```

. * etpoisson: ATE following ML for endogenous binary treatment
. margins, predict(cte) vce(unconditional)
Predictive margins                                         Number of obs = 1,993
Expression: Conditional treatment effect, predict(cte)

```

	Unconditional					
	Margin	std. err.	z	P> z	[95% conf. interval]	
_cons	3.754155	.3672285	10.22	0.000	3.034401	4.47391

The implied ATE is 3.754 visits compared with 2.316 in the Poisson model that treated private insurance as exogenous.

The ATET is obtained by restricting attention to those with private insurance.

```
. * etpoisson: ATET following ML for endogenous binary treatment
. margins, predict(cte) subpop(if private==1) vce(unconditional)
Predictive margins                                         Number of obs     = 1,993
                                                               Subpop. no. obs = 1,634
Expression: Conditional treatment effect, predict(cte)
```

	Unconditional					
	Margin	std. err.	z	P> z	[95% conf. interval]	
_cons	3.678841	.301032	12.22	0.000	3.088829	4.268853

25.4.5 Manual computation of TEs

When the `margins` command is not available as a postestimation command, the `predict` command can be used to calculate the TE “manually” as follows.

Consider an arbitrary variable x . Consider two possible values of interest, say, x^* and $x^* + \delta$. Using the `predict` command, we evaluate an in-sample prediction of the outcome variable y at two points $y(x = x^*)$ and $y(x = x^* + \delta)$. The ME is then defined by $\{y(x = x^* + \delta) - y(x = x^*)\}/\delta$. Averaging this over the full (or the treated) sample yields an estimate of ATE (or ATET).

As an illustration, we apply this method to compute the ATE in the current application.

```

. * etpoisson: Manual computation of ATE using predict command
. qui etpoisson docvis i.chronic i.female age,
>     treat(private= i.chronic i.female age income) vce(robust) nolog
. preserve
. qui replace private=1 if private==0
. qui predict docvis1 if private==1, pomean
. qui replace private=0 if private==1
. qui predict docvis0 if private==0, pomean
. generate ate = docvis1 - docvis0
. summarize ate docvis1 docvis0

      Variable   Obs    Mean    Std. dev.    Min    Max
ate          1,993  3.751386  3.228703  1.068299  12.34738
docvis1      1,993  4.929249  4.242453  1.403725  16.22422
docvis0      1,993  1.177863  1.013751  .3354255  3.876841

. restore

```

The ATE estimate of 3.751 differs slightly from 3.754 obtained earlier using the command `margins predict(cte)` but is identical to the estimate using the command `margins, r.private`. A bootstrap will yield the standard error of the ATE estimate.

25.4.6 The eteffects command

The `eteffects` command specifies separate potential outcomes for the treated and untreated, where the model has a nonlinear conditional mean and an additive error. The command has the following syntax:

```
eteffects (ovar omvarlist) [ , omodel noconstant ],
(tvar tmvarlist [ , noconstant ]) [if] [in] [weight] [ , stat options]
```

where *ovar* is the dependent variable of the outcome model, *omvarlist* is the list of exogenous indepvars in the outcome model, *tvar* is the binary ET variable, and *tmvarlist* is the list of covariates that predict treatment assignment.

The outcome model can be linear, probit, exponential mean, or fractional probit. The *stat* options are `ate`, `atet`, or `pomeans`.

For example, consider an exponential conditional mean. Then $y_{1i} = \exp(\mathbf{x}'_i \boldsymbol{\beta}_1) + u_{1i}$ and $y_{0i} = \exp(\mathbf{x}'_i \boldsymbol{\beta}_0) + u_{0i}$. Estimation is by a control function or residual augmentation approach. A probit model for the ET is specified, and the fitted residual \hat{v}_i is computed and is added as a regressor in the potential-outcome models. We then obtain OLS estimates of $y_{1i} = \exp(\mathbf{x}'_i \boldsymbol{\beta}_1 + \gamma_1 \hat{v}_i) + u_{1i}$ and $y_{0i} = \exp(\mathbf{x}'_i \boldsymbol{\beta}_0 + \gamma_0 \hat{v}_i) + u_{0i}$.

25.5 The LATE estimator for heterogeneous effects

In a classic RCT, the TE can be heterogeneous, varying from individual to individual, because estimating the ATE requires computation only of \bar{y}_1 and \bar{y}_0 . By contrast, the canonical model (25.1)–(25.3) specifies the same homogeneous TE α for all individuals. The regression-adjustment or potential-outcomes model (25.4) does allow for a limited form of heterogeneity. In the remainder of this chapter, we focus on methods that estimate aggregate TEs such as ATE when individual TEs are heterogeneous and only limited restrictions are placed on the nature of this heterogeneity.

In this section, we present a leading example, the LATE estimator. The basic IV or 2SLS estimate $\hat{\alpha}$ is usually interpreted as estimating the same TE for all individuals. LATE provides a reinterpretation when TEs are heterogeneous.

We provide a very brief summary of LATE. For further details, see [Angrist, Imbens, and Rubin \(1996\)](#), a lengthy presentation in [Angrist and Pischke \(2009, chap. 4\)](#), and, for example, [Cameron and Trivedi \(2005, chap. 25.7\)](#).

25.5.1 LATE with binary treatment and binary instrument

We consider a binary outcome y , binary treatment D , binary instrument z , and no additional control variables. A setting where this is particularly appropriate is where z equals one if randomly assigned to treatment in an RCT, D is the treatment indicator, but not all individuals assigned to treatment actually get the treatment.

We consider heterogeneous TEs. The effect of treatment on the outcome is $\Delta y / \Delta D$, which can be decomposed as $(\Delta y / \Delta z) / (\Delta D / \Delta z)$, leading to the TE

$$\text{TE}_{\text{Wald}} = \frac{E(y|z=1) - E(y|z=0)}{E(D|z=1) - E(D|z=0)} \quad (25.6)$$

This is the population analog of the Wald estimator. TE_{Wald} can be shown to equal $\text{Cov}(Z, y)/\text{Cov}(Z, D)$; see, for example, [Angrist and Pischke \(2009, 127\)](#). So TE_{Wald} can be consistently estimated by IV regression of y on D with instrument z . (These results do not restrict D to be binary.)

The IV estimator is usually interpreted as being the estimate of α in the homogeneous TES model $y_i = \beta + \alpha D_i + u_i$. In what follows, we simplify the expression in [\(25.6\)](#) when TES are heterogeneous.

For simplicity, throughout section [25.5](#) we consider the case where $z = 1$ on average pushes individuals toward treatment ($D = 1$) rather than toward nontreatment. Given binary D and z , the literature distinguishes between four types of individuals that exhaust all possible types. A complier may switch toward treatment if $z = 1$, a defier may switch away from treatment if $z = 1$, an always-taker always receives treatment regardless of the value of z , and a never-taker is always untreated regardless of the value of z . Table [25.2](#) provides formal definitions.

Table 25.2. LATE: The four types of individuals

Storage type	Bytes
Compliers (C)	$D = 1$ when $z = 1$ and $D = 0$ when $z = 0$
Defiers (D)	$D = 0$ when $z = 1$ and $D = 1$ when $z = 0$
Always-takers (A)	$D = 1$ regardless of the value of z
Never-takers (N)	$D = 0$ regardless of the value of z

Now consider the numerator in [\(25.6\)](#). Given the four possible types of individuals defined in table [25.2](#), we have

$$\begin{aligned}
E(y|z=1) - E(y|z=0) &= [\Pr(C)\{E(y|z=1, C) - E(y|z=0, C)\} \\
&\quad + \Pr(D)\{E(y|z=1, D) - E(y|z=0, D)\} \\
&\quad + \Pr(A)\{E(y|z=1, A) - E(y|z=0, A)\} \\
&\quad + \Pr(N)\{E(y|z=1, N) - E(y|z=0, N)\}] \\
&= [\Pr(C)\{E(y|z=1, C) - E(y|z=0, C)\} \\
&\quad + \Pr(D)\{E(y|z=1, C) - E(y|z=0, C)\}]
\end{aligned}$$

where the terms for always-takers and never-takers equal zero by definition. So the numerator confounds compliers and defiers.

A key component of the LATE interpretation of IV is to assume that there are no defiers. Then the numerator simplifies to a term involving only compliers,

$$E(y|z=1) - E(y|z=0) = \Pr(C)\{E(y|z=1, C) - E(y|z=0, C)\} \quad (25.7)$$

Furthermore, given no defiers, only compliers change to treatment status as z changes, so for binary D we have that the denominator in (25.6) equals the probability of being a complier,

$$E(D|z=1) - E(D|z=0) = \Pr(C) \quad (25.8)$$

Substituting (25.7) and (25.8) into (25.6) and canceling the common term $\Pr(C)$ yields

$$\text{TE}_{\text{Wald}} = E(y|z=1, C) - E(y|z=0, C)$$

The Wald estimand, which can be consistently estimated by IV, therefore equals the TE for compliers.

25.5.2 Assumptions for LATE

The assumptions underlying the LATE estimator are a combination of those for the potential-outcomes framework, those for IV estimation, and the assumption of no defiers.

Index potential outcomes y_0 and y_1 by treatment status $D = 0, 1$, and index potential treatment status D_0 and D_1 by the instrument $z = 0, 1$. We make the following assumptions:

1. Independence: (y_1, y_0, D_1, D_0) is jointly independent of z .
2. Exclusion: z does not directly determine potential outcomes y_1 and y_0 .
3. First stage: $E(D_1) \neq E(D_0)$.
4. Monotonicity: $\Pr(D_1 \geq D_0) = 1$.

The fourth assumption is replaced by $\Pr(D_0 \geq D_1) = 1$ if instead $z = 1$ on average pushes individuals toward nontreatment ($D = 0$). The assumption rules out defiers. It is called the monotonicity assumption because for nonbinary instrument z , it generalizes to the assumption that the probability of treatment is nondecreasing in z for all individuals.

Assumption three is straightforward to check from data, but the remaining assumptions are at best only partly testable. Some justification for these assumptions should be provided in any application.

25.5.3 Further discussion of LATE

LATE estimates a TE for compliers. One cannot identify whether an individual is a complier, so the ability to generalize LATE estimates is limited. However, one can identify the fraction of the sample that is compliers. Given monotonicity and the binary setup, (25.8) gives the expression for $\Pr(C)$, which can be estimated by first-stage regression of treatment D on the instrument z .

LATE does not give the ATET, because the treated group includes both compliers and always-takers. Similarly, LATE does not give the ATE for the untreated, because this group includes never-takers.

In practice, there may be more than one binary instrument. In that case, IV with different instruments leads to different LATE estimates because different

instruments lead to different groups of compliers. The 2SLS estimate using all instruments at once equals a weighted average of the individual LATE estimates.

The analysis so far does not include covariates. If the instrument is randomly assigned with all individuals having the same probability that $z = 1$, then there is no need for covariates. In other settings, the assumptions in section 25.5.2 may be reasonable only after conditioning on additional variables \mathbf{x} . Adding such variables may also improve estimator efficiency.

When the regressors \mathbf{x} take only a few distinct values, one can estimate by IV where the model for outcome y and the first-stage model for treatment D are fully saturated models with full sets of dummies for each discrete value of \mathbf{x} . The resulting estimated coefficient of D in the outcome equation is a weighted average of covariate-specific LATE estimates.

[Abadie \(2003\)](#) considers estimation of $E(y_i|D_i, \mathbf{x}_i, D_{1i} > D_{0i})$, the conditional mean of the outcome given treatment status and control variables with attention restricted to compliers. Compliers cannot be individually identified, but they can be in expectation. Using a linear model for the conditional mean as an approximation, [Abadie \(2003\)](#) proposes the weighted least-squares estimator that minimizes

$$S(\alpha, \beta) = \sum_{i=1}^N \hat{\kappa}_i (y_i - \alpha D_i - \mathbf{x}'_i \beta)^2$$

where the $\hat{\kappa}_i$ are inverse-probability weights, called kappa weights, that are consistent estimates of

$$\kappa_i = 1 - \frac{D_i \{1 - E(z_i|D_i, y_i, \mathbf{x}_i)\}}{1 - \Pr(z_i = 1|\mathbf{x}_i)} - \frac{\{(1 - D_i)E(z_i|D_i, y_i, \mathbf{x}_i)\}}{\Pr(z_i = 1|\mathbf{x}_i)} \quad (25.9)$$

The rationale for the weights is that given the monotonicity assumption, there are no defiers, so that we are left with compliers after subtracting away

always-takers, who have $D_{0i} = 1$ so $D_i(1 - z_i) = 1$, and never-takers, who have $D_{1i} = 0$ so $(1 - D_i)z_i = 1$. The expectation $E(z_i|D_i, y_i, \mathbf{x}_i)$, rather than the binary z_i , is used in (25.9) to ensure that the weights κ_i are positive.

[Angrist and Imbens \(2005\)](#) consider generalization to treatment D that takes several values. Then the 2SLS estimator is a weighted average of per-unit average causal effects for compliers.

25.5.4 Application of LATE

The LATE framework is especially applicable in RCTs where the offer of treatment (z) is randomly assigned but actual receipt of treatment (D) is voluntary. Then z is clearly a valid instrument for D and is a strong instrument unless exceptionally few people take up the offer of treatment.

We return to the Oregon Health Insurance Experiment studied in sections [24.8–24.9](#). A simple model regressed out-of-pocket expenditure (`oop`) on whether one wins or loses the lottery (`lottery`) and some lottery controls (`xlist`). It was found that winning the lottery was associated with lower out-of-pocket expenditure, an estimated effect that is an intention-to-treat effect.

Here we instead consider the effect of actual treatment, where the treatment variable `medicaid` equals one if the person actually enrolls in Medicaid. We have

```
. * LATE: Treatment D is Medicaid and instrument z is lottery win/lose  
. qui use mus224ohiesmallrecode, clear  
. label variable medicaid "Enrolled in Medicaid"
```

```
. tabulate medicaid lottery, cell nokey
```

Enrolled in Medicaid	Selected in the lottery		Total
	Not selec	Selected	
Not enrolled	9,873 43.53	6,428 28.34	16,301 71.88
Enrolled	1,530 6.75	4,848 21.38	6,378 28.12
Total	11,403 50.28	11,276 49.72	22,679 100.00

Have 49.7% of the sample was selected by the lottery, and 28.1% enrolled in Medicaid. Note that 6.75% of the sample was in Medicaid, even though they were not selected in the lottery. The monotonicity assumption of no defiers means that all of these people are assumed to be always-takers.

More than one lottery occurred, and lotteries varied with some household characteristics. These variables are included as controls in the global `xlist`. We therefore regress `oop` on `medicaid` and `controls` `xlist`, where `medicaid` equals one if the person actually enrolls in Medicaid. The LATE estimate is obtained by 2SLS estimation with `lottery` an instrument for `medicaid`. We obtain the following estimates.

```

. * LATE: Treatment is Medicaid and instrument is lottery win/lose
. global y oop // Outcome variable for this chapter
. global xlist dhssize2 dhssize3 dlotdraw* dsurvdraw* // x variables for treat
. qui regress $y lottery $xlist, vce(cluster household_id)
. estimates store intent
. qui regress $y medicaid $xlist, vce(cluster household_id)
. estimates store ols
. qui regress medicaid lottery $xlist, vce(cluster household_id)
. estimates store first
. qui ivregress 2sls $y (medicaid = lottery) $xlist, vce(cluster household_id)
. estimates store iv
. estimates table intent ols first iv, keep(medicaid lottery)
>     b(%10.3f) se stat(N r2)

```

Variable	intent	ols	first	iv
medicaid		-170.331 9.177		-137.074 33.703
lottery	-40.924 10.130		0.299 0.006	
N	22679	22679	22679	22679
r2	0.002	0.012	0.114	0.011

Legend: b/se

The first column of the output gives the intention-to-treat effect, replicating the section [24.9](#) result that being selected by the lottery leads to a \$40.92 reduction in out-of-pocket expenditures. The second column provides the potentially inconsistent OLS estimate of a \$170.33 reduction in out-of-pocket expenditures for those enrolled in Medicaid. The third column gives the first-stage regression estimate and implies that the complier group is 29.9% of the sample.

The final column shows the preferred LATE estimate of a \$137.07 reduction in out-of-pocket expenditures, with a standard error of \$33.70. This is a very large TE because the sample average of `oop` is \$269.01. When the control variables are dropped, the LATE estimate becomes – 151.01 with a standard error of 33.33.

In summary, the causal effect of Medicaid enrollment is to reduce out-of-pocket expenditures on average by \$137 for the 30% of the sample that is compliers.

25.5.5 Marginal TEs

The LATE framework has been extended to define the marginal treatment effect (MTE) function; see, for example, [Heckman and Vytlacil \(2005\)](#).

A treatment selection equation $D = \mathbf{1}(\gamma z + \mathbf{x}'\gamma_2 + v > 0)$ can be rewritten as $\Pr\{p(z, \mathbf{x}) > U_D\}$ where $p(\cdot)$ is the propensity score $\Pr(D = 1|z, \mathbf{x})$ and U_D is uniformly distributed on $(0, 1)$. U_D is interpreted as an individual's unobserved (after conditioning on z and \mathbf{x}) propensity or indifference or resistance to treatment.

The MTE(\mathbf{x}, u) is then the ATE conditional on \mathbf{x} , as usual, and additionally on $U_D = u$.

The community-contributed `mtefe` command ([Andresen 2018](#)) allows estimation of the distribution of various TEs fitting both parametric and semiparametric MTE models.

The MTE can be decomposed as the difference in two potential-outcome functions: $E(y_1|\mathbf{x}, u)$ and $E(y_0|\mathbf{x}, u)$. The community-contributed `mtebinary` command ([Kowalski, Tran, and Ristovska 2016](#)) uses this distinction to enable finer distinction between always-takers, compliers, and never-takers.

There is a tension between a structural approach that requires strong assumptions and quasi-experimental approaches such as LATE that require fewer assumptions but are not as generalizable. [Heckman \(2010\)](#) critiques LATE using the MTE framework. [Imbens \(2010\)](#) provides a response that details what can be learned from quasi-experimental methods such as LATE.

25.6 Difference-in-differences and synthetic control

The fixed-effects (FE) estimator for linear panel-data models controls for endogeneity by assuming that the error can be decomposed as $\alpha_i + u_{it}$ and that the endogenous regressor is correlated only with the time-invariant component α_i . Mean-differencing or first-differencing eliminates α_i and enables consistent estimation.

This approach requires repeated measures for each individual over time. The DID method does not require repeated measures but does add the assumption of parallel trends; see the lengthy introductory presentation in section 4.8. In this section, we summarize some extensions.

25.6.1 DID

A common application of the DID method is to settings where individual-level data are available over time but the binary treatment (D) occurs at a more aggregate level, such as state or village.

Then we consider the two-way fixed-effects model

$$y_{ist} = \phi_s + \gamma_t + \alpha D_{st} + \mathbf{x}'_{ist} \boldsymbol{\beta} + u_{ist}, \quad i = 1, \dots, N \quad (25.10)$$

where s denotes state, t denotes time, i denotes an individual who, unlike the panel case, is observed only once (that is, only in one particular state and time period), and \mathbf{x}_{ist} are exogenous control variables. The parallel trends assumption is that separate state fixed effects ϕ_s and time fixed effects γ_t are adequate, rather than more general state-time fixed effects. Interest lies in estimating the ATET parameter α .

OLS estimation of (25.10) using the `regress` command is straightforward with inference based on standard errors clustered at the state level.

Alternatively, the `didregress` command, introduced in Stata 17, fits this model and some variants and additionally provides some diagnostics and

better methods to compute standard errors when there are few clusters (here states).

The `didregress` command has syntax

```
didregress (ovar omvarlist) (tvar[, continuous]) [if] [in] [weight], group(groupvars)  
[time(timevar) options]
```

Here *ovar* is the outcome, *omvarlist* are control variables, *tvar* is the treatment variable that can be binary (the default) or continuous (option *continuous*), *groupvars* are the grouping variables, and *timevar* is the time variable (should there be one).

For the model (25.10), there is a single group variable (state) and a time variable. An alternative DID example may use contrast across two distinct groups, such as state and gender, rather than state and time. Then two *groupvars* are provided, and the *time()* option is dropped. For difference in difference in differences an additional *groupvar* is provided.

The related `xtdidregress` command can be used when panel data are available, rather than repeated cross-sections. Then the group-specific estimate effect ϕ_s in (25.10) is replaced by an individual-specific effect ϕ_i .

The default standard errors are cluster–robust standard errors with clustering at the group level. The `didregress` command controls for group-specific effects using the `areg` command. The `xtdidregress` command controls for group-specific effects using the `xtreg, fe` command. Thus, these two commands use different formulas for degrees-of-freedom adjustments in computing cluster–robust standard errors. The `areg` formula leads to larger standard errors, and we favor use of the `xtreg, fe` formula; see section 6.6.4 for a detailed discussion.

A common complication in DID analysis is that there can be few clusters, or few treated clusters, in which case cluster–robust standard errors lead to tests with the wrong size and with the wrong coverage. The `wildbootstrap` option provides *t* statistics and confidence intervals based on the wild cluster bootstrap presented in section 12.6. The option `vce(hc2)` bases inference on an alternative bias-corrected variance–covariance matrix of the estimator, and the `dlang` option uses the aggregated data of [Donald and Lang \(2007\)](#).

Developing better inference methods in this setting remains an active area of research.

A key assumption of the DID method is the parallel trends assumption that γ_t in (25.10) does not vary by treatment status. The `estat trendplots` postestimation command provides a graphical diagnostic of the parallel trends assumption, and the `estat ptrends` postestimation command provides a test of linear parallel trends. The `estat granger` postestimation command provides a test of whether a TE occurs before the actual time of treatment. The `estat grangerplot` command plots coefficients from leads and lags of the treatment indicator variable.

Some DID applications have differential timing of when treatment occurs. [Goodman-Bacon \(2018\)](#) shows that the DID estimator is then a weighted average of all possible two-group two-period DID estimators. [Callaway and Sant'Anna \(2021\)](#) propose separate estimation for each distinct treatment, followed by aggregation to obtain more precise estimates. [Wooldridge \(2021\)](#) proposes a two-way Mundlak estimator.

[Borusyak, Jaravel, and Spiess \(2022\)](#) propose an imputation estimator that is efficient under suitable assumptions and contrast this estimator with other recently proposed estimators.

The DID methodology is quite flexible and can be extended beyond the setting considered here. Some extensions for nonlinear regressions are covered in [Athey and Imbens \(2006\)](#). Other extensions from the perspective of propensity scores and matching methods are covered in [Lechner \(2011\)](#). [Huber \(2019\)](#) provides many references. [Miller \(2021\)](#) reviews the closely related subject of event study models.

25.6.2 Synthetic control

Synthetic control methods measure the TE for a treated unit as the difference between the outcome for the treated observation and a single counterfactual (untreated) value of the outcome. The counterfactual, called a synthetic control, is a weighted average of the outcomes for untreated observations. The weights are selected so that the synthetic control is similar to the treated unit in a pretreatment period.

An advantage of synthetic control compared with DID is that the parallel trends assumption that changes over time are the same for treated and untreated individuals, after controlling for group fixed effects and regressors, can be relaxed. Synthetic control essentially allows for an interactive effect $\phi_s \gamma_t$ in (25.10). Statistical inference, however, is challenging.

We illustrate synthetic control, with little explanation of the underlying theory, using the example of [Abadie, Diamond, and Hainmueller \(2010\)](#) who estimated the effect on cigarette consumption of a large-scale tobacco control program instituted in California in 1988.

The analysis is based on the following panel dataset:

- . * Synthetic control: Smoking dataset and summary
- . qui use mus225smoking, clear
- . summarize, sep(7)

Variable	Obs	Mean	Std. dev.	Min	Max
state	1,209	20	11.25929	1	39
year	1,209	1985	8.947973	1970	2000
cigsale	1,209	118.8932	32.7674	40.7	296.2
lnincome	1,014	9.861634	.1706769	9.397449	10.48662
beer	546	23.4304	4.22319	2.5	40.4
age15to24	819	.175472	.0151589	.1294482	.2036753
reprice	1,209	108.3419	64.38199	27.3	351.2

The panel is one of 39 states observed over the 31 years from 1970–2000. Some variables are not observed in all years.

Synthetic control methods compare a treated unit, here California, with a control unit that is a weighted sum of the untreated units. Let s denote a state where $s = 1$ for the treated state and $s = 2, \dots, J + 1$ for the J untreated states. And let $t = 1, \dots, T_0, T_0 + 1, \dots, T$, where treatment occurs between T_0 and $T_0 + 1$.

Time-invariant weights w_s , $s = 2, \dots, J + 1$, are selected so that in each year of the pretreatment period, a weighted combination of untreated states has similar outcome to that in the treated state: $y_{1t} \simeq \frac{1}{J} \sum_{s=2}^{J+1} w_s y_{st}$, $t = 1, \dots, T_0$. These weights that sum to one are determined by an algorithm given in [Abadie, Diamond, and Hainmueller \(2010\)](#), for example, that uses

data on the outcome and control variables in the pretreatment period. Typically, the weights are zero for all but a few untreated units.

Given these weights, the estimated TE is simply the posttreatment period difference between the outcome for California and the weighted average of the outcome in the untreated states.

$$\widehat{\text{TE}}_{\text{synth}} = y_{1t} - \frac{1}{J} \sum_{s=2}^{J+1} w_s y_{st}, \quad t = T_0 + 1, \dots, T$$

For the current example, the variables used to determine the weights are beer consumption per capita (`beer`), log state income per capita (`lnincome`), retail price of cigarettes (`retprice`), and percent of state population aged 15–24 years (`age15to24`). These variables are used as averages over the entire pretreatment period or that part of the pretreatment period for which they are available. Additionally, the outcome `cigsale` in the selected years 1975, 1980, and 1988 is used.

The community-contributed `synth` command ([Abadie, Diamond, and Hainmueller 2014](#)) obtains the weights. The option `trunit(3)` identifies the treated state—California appears as the third state in the panel. The option `trperiod(1989)` defines the posttreatment period as beginning in 1989. We have

```
. * Synthetic control: synth command
. tsset state year
Panel variable: state (strongly balanced)
Time variable: year, 1970 to 2000
    Delta: 1 unit
. qui synth cigsale beer(1984(1)1988) lnincome(1972(1)1988) retprice
>     age15to24 cigsale(1988) cigsale(1980) cigsale(1975),
>     trunit(3) trperiod(1989) figure
```

The lengthy output from the command is omitted. It includes the weights that are 0 for all states except Colorado (0.285), Connecticut (0.101), Nevada (0.245), and Utah (0.369). The option `figure` gives a plot identical to the first panel of figure [25.1](#).

The community-contributed `synth_runner` package ([Galiani and Quistorff 2017](#)) is an extension to the `synth` command that provides useful graphs and statistics. We have

```
. * Synthetic control: synth_runner command
. synth_runner cigsale beer(1984(1)1988) lnincome(1972(1)1988) retrice
>     age15to24 cigsale(1988) cigsale(1980) cigsale(1975),
>         trunit(3) trperiod(1989) gen_vars
Estimating the treatment effects
Estimating the possible placebo effects (one set for each of the 1 treatment
> periods)
|                               | Total: 38
.....| 20.00s elapsed.

Conducting inference: 5 steps, and 38 placebo averages
Step 1... Finished
Step 2... Finished
Step 3... Finished
Step 4... Finished
Step 5... Finished

Post-treatment results: Effects, p-values, standardized p-values
```

	estimates	pvals	pvals_std
c1	-7.887098	.1315789	0
c2	-9.693599	.1842105	0
c3	-13.8027	.2105263	0
c4	-13.344	.1315789	0
c5	-17.0624	.1052632	0
c6	-20.8943	.0789474	0
c7	-19.8568	.1315789	.0263158
c8	-21.0405	.1578947	0
c9	-21.4914	.1052632	.0263158
c10	-19.1642	.1842105	.0263158
c11	-24.554	.1052632	0
c12	-24.2687	.1052632	.0263158

The first column of the output gives the estimated TES in each year. For example, the causal effect of the tobacco-control program is estimated to have reduced cigarette sales by 24.27 packs 12 years after the intervention. This is a very large effect because the sample average was 118.89 packs. The remaining columns of the output will be explained below.

The postestimation command `effect_graphs` produces two plots that compare the outcome over time for the treated unit with that for the synthetic control.

```

. * Synthetic control: synth_runner postestimation command effect_graphs
. effect_graphs, trlinediff(-1) tc_options(ti("Treated & control outcomes"))
>     effect_options(title("Difference between treated & control"))
. graph combine tc effect, xcommonysize(2.5) xsize(6) scale(1.4)

```

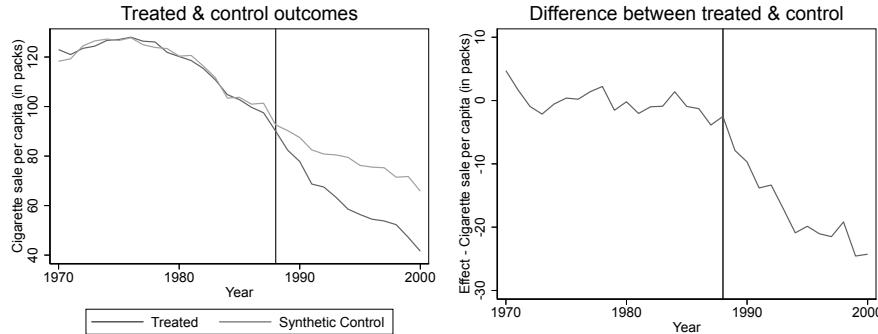


Figure 25.1. Synthetic control: Outcomes and difference in outcomes

The first panel of figure 25.1 plots over time the outcome for California and for the synthetic control. The two are very similar in the pretreatment period, as expected because the weights are selected to ensure this. In the posttreatment period, the two diverge. The second panel of figure 25.1 plots the difference in the curves. For example, the endpoint in year 2000 is an estimated reduction of approximately 25; more precisely, this is the estimate of -24.27 reported in the output from the `synth_runner` command.

The postestimation command `single_treatment_graphs` produces two plots that compare the outcome over time for the treated unit with those for each of the untreated units.

```

. * Synthetic control: synth_runner postestimation command single_treatment_graphs
. single_treatment_graphs, trlinediff(-1) effects_ylabels(-50(10)50)
>     do_color(gs11) effects_ymax(50) effects_ymin(-50)
>     raw_options(title("Treated & donor outcomes"))
>     effects_options(title("Difference between treated & donor outcomes"))
(6 real changes made)
(0 real changes made)
. graph combine raw effects, xcommonysize(2.5) xsize(6) scale(1.3)

```

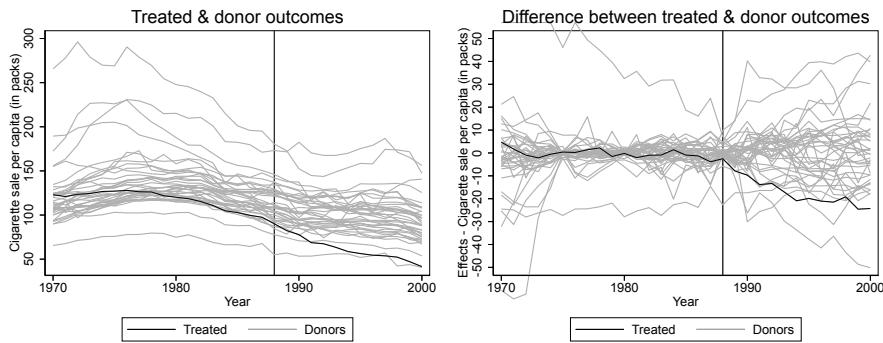


Figure 25.2. Synthetic control: Treated and donor outcomes

The first panel of figure 25.2 plots the outcomes over time for California (the solid line) and for all 38 other states.

The solid line for California in the second panel of figure 25.2 is the same as that in the second panel of figure 25.1. The remaining curves are placebo curves where the synthetic control method is applied separately to each untreated state (with California omitted in forming the synthetic control).

Considering the final year of 2000, for example, from the second panel of figure 25.2, one of the 38 placebos has a response less than -24.27 , the California response, and 3 have a response that exceeds -24.27 . If treatment was randomly assigned, then by a two-sided permutation test, the p -value is $4/38 = 0.10526316$. This is the value given in the last entry in the second column of the output from the `synth_runner` command. The third column provides an alternative p -value that divides posttreatment effects for all units by pretreatment match quality as measured by root mean squared prediction error.

In fact, treatment is not randomly assigned, but statistical inference for synthetic control under the alternative of nonrandom assignment is challenging. For robustness and diagnostic tests, and extensions to more than one treated state, see, for example, the recent detailed survey by [Abadie \(2021\)](#).

25.7 Regression discontinuity design

A natural experiment in econometrics is an observational setting, rather than a controlled experiment setting, in which an intervention is observed that is strictly exogenous and elicits a causal response in a variable of interest. If the intervention can be validly interpreted as “treatment”, then the nontreatment observations serve as a counterfactual, making possible inference about the causal impact of the intervention. A key feature of a natural experiment is that even though the intervention does not follow any experimental design, treated and nontreated samples are generated analogously to a randomized trial. Then a parameter such as ATE or ATET can be identified and estimated.

RD design is a leading example of a natural experiment that is generated by the existence of a discontinuous treatment assignment based on a nonmanipulable threshold. An example is enrollment eligibility into the U.S. Medicare public health insurance program, which occurs when an individual crosses the threshold age of 65. One can estimate the average impact on healthcare use of Medicare eligibility by comparing samples of individuals on either side of the age 65 threshold, though this comparison needs to also control for the complication that healthcare use itself varies considerably with age.

RD methods were originally proposed in the educational psychology literature as an alternative to RCT. [Hahn, Todd, and Van der Klaauw \(2001\)](#) provide the underlying theory. A key assumption, which is empirically testable, is that close to the point of discontinuity, also called the cutoff, the treated and untreated individuals are well matched in their observed characteristics. The estimated TE then applies to those who are close to the cutoff point; that is, the effect is “local” and possibly not capable of extrapolation to a larger and more inclusive population.

In this section, we focus mainly on sharp regression discontinuity (SRD), the simplest and leading type of RD design, and use of the `rdrobust` package of [Calonico, Cattaneo, and Titiunik \(2014\)](#) and [Calonico et al. \(2017\)](#). A detailed presentation of RD methods is given in [Lee and Lemieux \(2010\)](#).

25.7.1 Sharp regression discontinuity design

In an RD design, treatment status is determined by the value taken by a so-called running variable or forcing variable x_i . Under SRD, there is a known nonmanipulable cutoff c such that no treatment occurs below the threshold, while all cases that cross the threshold receive treatment. Thus, the treatment variable $D_i = \mathbf{1}(x_i \geq c)$, so that $\Pr(D_i = 1|x_i \geq c) = 1$ and $\Pr(D_i = 1|x_i < c) = 0$. This rule determining treatment is independent of individual characteristics, and hence by design, there is no idiosyncratic or selection element in treatment. All of those selected for treatment are compliers.

Interest lies in the effect of treatment on an outcome variable y_i . Using potential-outcome notation for a binary treatment, we are interested in estimating the ATE $E[y_{1i} - y_{0i}]$. For identification of the TE, additional assumptions are needed. To rule out the possibility of confounding, we assume that there are no other possible sources of discontinuity in response such as a discontinuity in the running variable x or a discontinuity in the density function of the random-error term on the regression. Both sources of discontinuity must be ruled out by assumption.

If TEs are homogeneous, then the estimated ATE is the estimate of α from the OLS regression

$$y_i = \phi + \alpha D_i + f(x_i) + u_i, \quad i = 1, \dots, N$$

where $f(x_i)$ is a specified function of the running variable, such as a polynomial in x_i . A richer model adds interaction terms $D_i \times f(x_i)$.

This parametric approach uses all the data but has the limitation of requiring correct specification of the functional form for $E(y_i|D_i, x_i)$.

An alternative local approach considers only outcomes in the neighborhood of the threshold value c . Then,

$$\text{ATE}_{\text{SRD}} = y^+ - y^-$$

where $y^+ = \lim_{x \rightarrow c^+} E(y_i|x_i = c)$ and $y^- = \lim_{x \rightarrow c^-} E(y_i|x_i = c)$. This local approach allows for heterogeneous TES.

A simple local approach compares sample means of y on either side of the threshold, but this fails to control for the fact that y varies with x in general and not just at c . The preferred approach is to instead use a local linear regression on either side of the threshold. This requires choice of how local to be, essentially choice of a kernel bandwidth, and can mean a loss of efficiency because only a subset of the data is being used.

Under the assumption of local randomization of treatment assignment, conditional independence holds so that $y_{1i}, y_{0i} \perp D_i$. It is therefore important that the threshold c be nonmanipulable. For example, if the running variable is the number of employees in a firm and government regulations favor firms with at most 20 employees, then manipulation may occur, leading to a bunching of firms at or just below 20 employees.

25.7.2 SRD numerical and graphical illustration

This subsection provides a numerical illustration of RD concepts and issues pertaining to the estimation of TES. Subsequent subsections use the more specialized community-contributed commands `rdplot` and `rdrobust` to obtain graphs and estimates.

A sample of size 500 is generated using a regression model for outcome y that is a quadratic function of a running variable x . The data-generating process (DGP) for y is subject to a single discontinuity at the midpoint of the sample where $x = 0$ because of the treatment assignment variable D that takes the value 1 if $x \geq 0$ and value 0 if $x < 0$. For this DGP, the homogeneous TE is 80, with $y^- = -10$ and $y^+ = 70$.

```

. * SRD DGP: Quadratic in running variable x and TE 80 at x = 0
. clear all
. qui set obs 500
. set seed 10101
. generate x = 0.25*(_n - 250) + rnormal(0,5) // The running variable
. generate xsq = x^2
. generate D = x > 0 // The sharp cutoff
. generate y = -10 + 80*D + 2*x - 0.025*xsq + rnormal(0,60) // The outcome
. summarize y x D

```

Variable	Obs	Mean	Std. dev.	Min	Max
y	500	-2.063917	126.896	-344.0993	250.661
x	500	.1132601	36.33044	-71.85318	69.72129
D	500	.5	.5005008	0	1

RD methods make extensive use of graphical tools because these provide a good initial feel for the presence of a significant discontinuity. Skipping the intermediate step of formally testing for discontinuity in x at the cutoff $x = 0$, we next display the data as a scatter diagram that also displays quadratic regressions (and associated 95% confidence intervals) fitted separately to the treated and nontreated observations.

```

. * SRD design example: Scatterplot with separate global quadratic fits
. twoway (scatter y x, xline(0) yline(-10, lpat(dash)) yline(70, lpat(dash))
>     msize(vsmall) xtitle("Running variable x") ytitle("Outcome") leg(off))
>     (qfitci y x if D==1, lcolor(black)) (qfitci y x if D==0, lcolor(black)),
>     title("RD: Scatterplot and global quadratic fits")

```

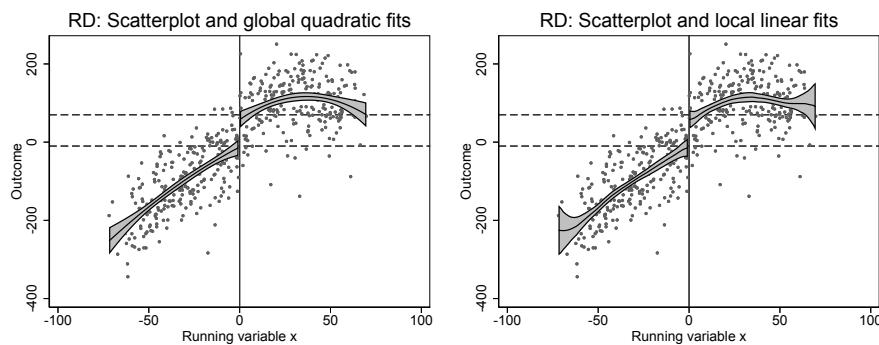


Figure 25.3. SRD: Scatterplot and global or local regression fits

The resulting graph is given in the first panel of figure 25.3. The two horizontal dashed lines plot the DGP values $y^- = -10$ and $y^+ = 70$. The

vertical line plots the discontinuity at $x = 0$. The estimated TE is the vertical difference between the two fitted curves at $x = 0$.

The following regression fits the same two quadratic curves and estimates the TE.

Linear regression							
		Number of obs	=	500			
	F(5, 494)	=	338.35				
	Prob > F	=	0.0000				
	R-squared	=	0.7738				
	Root MSE	=	60.656				

y	Coefficient	Robust std. err.	t	P> t	[95% conf. interval]	
D	70.17662	15.70607	4.47	0.000	39.31769	101.0355
x	2.415727	.7527344	3.21	0.001	.9367714	3.894683
xsq	-.0128546	.0116833	-1.10	0.272	-.0358096	.0101004
Dx	.7923694	1.128322	0.70	0.483	-1.424533	3.009272
Dxsq	-.030622	.0172868	-1.77	0.077	-.0645867	.0033427
_cons	-12.61842	10.84235	-1.16	0.245	-33.92122	8.684382

The estimated TE is 70.18 and is quite precisely estimated.

The first panel of figure 25.3 used global quadratic regression functions. A nonparametric regression is the preferred alternative because it avoids functional form assumptions that could potentially distort even informal inference about the cutoff. The following code instead fits the regression curves using local linear nonparametric regression, introduced in section 2.6.6, with a triangular kernel and plugin choice of bandwidth.

```
. * SRD design example: Scatterplot with separate local linear fits
. twoway (scatter y x, xline(0) yline(-10, lpat(dash)) yline(70, lpat(dash))
>     msize(vsmall) xtitle("Running variable x") ytitle("Outcome") leg(off))
>     (lpolyci y x if D==1, kernel(triangle) bw(20) deg(1) lcolor(black))
>     (lpolyci y x if D==0, kernel(triangle) bw(20) deg(1) lcolor(black)),
>     title("RD: Scatterplot and local linear fits")
```

The second panel of figure 25.3 gives the resulting graph. The local linear curves are very similar to the quadratic fits, aside from at the lower and upper endpoints. Interest lies in the fitted curves at $x = 0$. Visually, the nonparametric

estimates are similar to the quadratic estimates, being slightly below -10 at $x = 0$ for the left curve and slightly below 70 for the right curve. The nonparametric estimates at $x = 0$ are less precise than the parametric quadratic estimates because the 95% confidence intervals at $x = 0$ are wider.

The ATE estimate is the first fitted value for observations with $x \geq 0$ less the final fitted value for observation with $x < 0$, here the 50th fitted value because the default for `lpoly` is to fit at $\min(N, 50)$ points. The ATE estimate can be obtained as follows:

```
. qui lpoly y x if D==0, kern(tri) bw(20) deg(1) gen(xminus yminus)
. qui lpoly y x if D==1, kern(tri) bw(20) deg(1) gen(xplus yplus)
. di "ATE = " yplus[1] " - " yminus[50] " = " yplus[1]-yminus[50]
ATE = 58.223343 - -14.019219 = 72.242562
```

Implementation of this nonparametric approach requires choice of kernel and choice of bandwidth and a method to calculate the standard error of the estimated ATE. The subsequent application uses community-contributed commands that automate this process.

An alternative and popular nonparametric method for viewing the relationship between the outcome y and the running variable x is to form bins of x and calculate the mean of y in each bin. The following provides a scatterplot of the binned estimates, using 20 equal-spaced bins on each side of the treatment threshold, as well as the same local linear curves as those already given in the second panel of figure [25.3](#).

```
. * SRD design example: Binned data with separate local linear fits
. sum x
Variable |       Obs        Mean      Std. dev.       Min       Max
          x |     500    .1132601     36.33044   -71.85318   69.72129
. scalar lowbw = (0 - r(min))/20
. scalar highbw = (0 + r(max))/20
. generate xbin = lowbw*floor(x/lowbw)+ lowbw/2
. replace xbin = highbw*floor(x/highbw)+ highbw/2
(500 real changes made)
. bysort xbin: egen ybinmean = mean(y)
. twoway (scatter ybinmean xbin, xline(0) xtitle("Bins of running variable x"))
>           msiz(vsmal) ytitle("Mean of outcome in each bin of x") legend(off)
>           (lpoly y x if D==1, kern(tri) bw(20) deg(1) lcol(black) fcol(none))
>           (lpoly y x if D==0, kern(tri) bw(20) deg(1) lcol(black) fcol(none)),
>           title("RD: Scatterplot of binned data")
```

The first panel of figure 25.4 plots the fitted means and the fitted local linear curves. The outcome seems fairly continuous in the running variable, aside from the clear discontinuity in y at $x = 0$. An alternative to equal-spaced bins is to base the bins on quantiles of x .

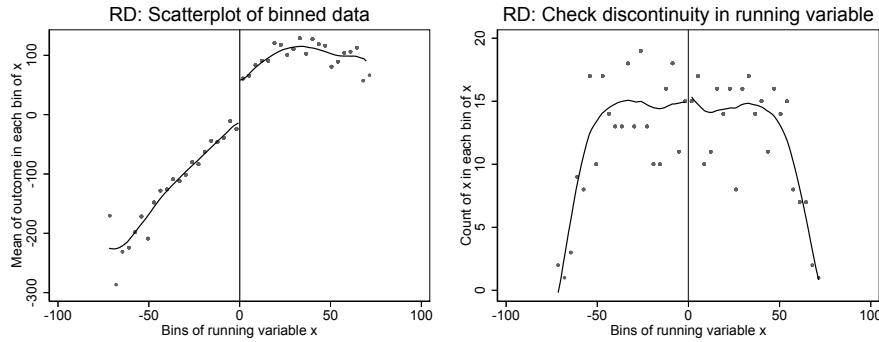


Figure 25.4. RD: Binned data scatterplot and check of discontinuity in the running variable

An important assumption is that there is no discontinuity in the running variable around the threshold value of the running variable. A visual check is to again form equal-spaced bins of x , calculate the number of observations in each bin, plot these counts and an associated fitted nonparametric curve against x , and check for a discontinuity at the threshold value of x . We have

```
. * SRD design example: Visual check no discontinuity in x at x = 0
. bysort xbin: egen xcount = count(x)
. twoway (scatter xcount xbin, xline(0) xtitle("Bins of running variable x"))
>     msize(vsmal) ytitle("Count of x in each bin of x") legend(off)
>     (lpoly xcount xbin if D==1, kern(triangle) bw(20) deg(1) lcol(black))
>     (lpoly xcount xbin if D==0, kern(triangle) bw(20) deg(1) lcol(black)),
>     title("RD: Check discontinuity in running variable")
```

The second panel of figure 25.4 suggests there is no discontinuity in the counts of x at the threshold $x = 0$. [McCrary \(2008\)](#) provides a formal nonparametric test based on this approach.

25.7.3 rdrobust package and application

The community-contributed `rdrobust` package of [Calonico, Cattaneo, and Titiunik \(2014\)](#) and [Calonico et al. \(2017\)](#) provides nonparametric estimates for various RD designs. There are three components. The `rdplot` command

provides scatterplots of y against x with fitted global polynomial curves. The `rdrobust` command provides nonparametric estimates of the ATE using local polynomial regression, where bandwidths are data determined, and provides heteroskedastic–robust or cluster–robust standard errors for this estimate. A key component is determining the bandwidth of the local regression. The `rdbwselect` command provides a range of methods for determining bandwidths that involve a tradeoff between variance that decreases as bandwidth increases and bias that increases as bandwidth increases.

These commands provide a very wide range of options, and users of these commands should read at least the two articles cited above. In the remainder of this section, we merely provide simple examples of these commands, with application to the same dataset as that used by [Calonico et al. \(2017\)](#).

The application is to U.S. Senate elections from 1914 to 2010. The running variable `margin` is the Democratic Party’s margin of victory in a U.S. Senate seat in year t , and the outcome variable `vote` is the Democratic Party’s share of the vote in the subsequent election for the same seat, an election that usually occurs in year $t + 6$. The hypothesis under consideration is that there is an incumbent advantage, so that a narrow win (loss) in one election is likely to lead to a win (loss) in the subsequent election. This is an SRD design with cutoff at `margin = 0`.

Unlike many panel-data applications, the estimators used here are pooled estimators without fixed effects. Also, heteroskedastic–robust standard errors, reported here, are similar to cluster–robust standard errors. See section [25.7.8](#) for discussion.

```
. * SRD: Cattaneo et al. U.S. Senate elections data
. qui use mus225rdsenate, clear
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
state	1,390	40.01367	21.99304	1	82
year	1,390	1964.63	28.05466	1914	2010
vote	1,297	52.66627	18.12219	0	100
margin	1,390	7.171159	34.32488	-100	100
class	1,390	2.023022	.8231983	1	3
termshouse	1,108	1.436823	2.357133	0	16
termssenate	1,108	4.555957	3.720294	1	20
population	1,390	3827919	4436950	78000	3.73e+07

The dataset includes additional variables, some of which are used below in validity tests.

25.7.4 The `rdplot` command

The community-contributed `rdplot` command provides a visual diagnostic that plots a variable y against the running variable x . If y is an outcome felt to be affected by the RD treatment, then the only discontinuity between y and x should be at the threshold value of x . If y is instead a variable unaffected by the RD treatment, such as a pretreatment variable, then there should be no discontinuity at the threshold value of x .

The graph produced by `rdplot` has two components—a scatterplot of bin means of y against x and global polynomial regression curves fit separately on either side of the threshold value of x .

The general syntax for the `rdplot` command is

```
rdplot depvar runvar [if] [in] [, options]
```

The option `c()` specifies the cutoff or threshold value of the running variable, with default `c(0)`. The option `p()` specifies the degree of the global polynomial, with default a fourth-degree polynomial. The option `ci()` adds pointwise confidence intervals for each bin, and the option `shade` shades these. The remaining options mostly determine the associated scatterplot that depends on the number of bins, the spacing of the bins, and the way that bin (conditional) means are computed.

The option `binselect()` provides eight different data-driven methods for determining the number of bins based on three underlying binary choices. The bins of x may be either equal spaced or quantile spaced. The bin means may be calculated using either spacing estimators or local polynomial estimators with the `kern()` option, allowing uniform (the default), triangle, or Epanechnikov kernels. A spacing estimator is based on ordered data within each bin and does not require any tuning parameters; see [Calonico, Cattaneo, and Titiunik \(2015\)](#). The number of bins is chosen to either minimize integrated mean squared error (MSE) or lead to binned sample means that have variability approximately equal to the amount of variability of the raw data. The latter method, termed “mimicking variance”, leads to undersmoothing and greater variability than

integrated MSE. The default `binselect(esmv)` uses spacing estimators of the mean in equally spaced bins whose number is chosen by the mimicking-variance methods. The option `nbins()` instead directly specifies the number of bins.

We begin by providing a scatterplot of the relationship between the running variable `margin`, the Democratic Party's margin of victory in an election for a U.S. Senate seat, and the preceding election (six years earlier) for the same seat.

```
. * SRD: Scatterplot of the data
. twoway scatter vote margin, msize(tiny) xline(0)
>          ytitle("Vote share at t+6") xtitle("Margin of victory at t")
>          title("Simple scatterplot of the data")
```

The first panel of figure 25.5 shows that a higher margin of victory in one election leads on average to a higher vote share in the subsequent election. What is not clear, however, is whether there is an incumbency advantage that leads to a discontinuity at `margin = 0`, the threshold for election victory by the Democratic candidate.

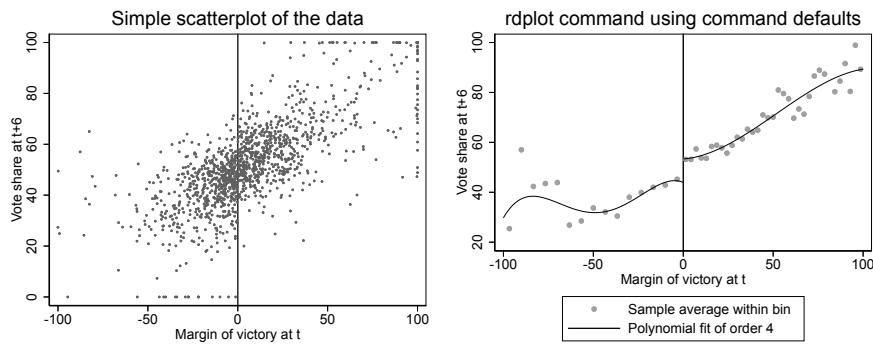


Figure 25.5. SRD: Scatterplot and binned plot

The `rdplot` command, with command defaults, for the same data yields the following results. The associated graph is given in the second panel of figure 25.5.

```

. * SRD: rdplot command for the same data using command defaults
. rdplot vote margin, c(0)
> graph_options(title("rdplot command using command defaults")
>                 ytitle("Vote share at t+6") xtitle("Margin of victory at t"))
RD Plot with evenly spaced mimicking variance number of bins using spacings
> estimators.

      Cutoff c = 0 | Left of c  Right of c
Number of obs      | 595        702
Eff. Number of obs | 595        702
Order poly. fit (p)| 4           4
BW poly. fit (h)  | 100.000    100.000
Number of bins scale | 1.000     1.000

Number of obs      = 1297
Kernel             = Uniform

Outcome: vote. Running variable: margin.

      | Left of c  Right of c
Bins selected      | 15          35
Average bin length | 6.667       2.857
Median bin length  | 6.667       2.857

IMSE-optimal bins | 8           9
Mimicking Var. bins | 15         35

Rel. to IMSE-optimal:
    Implied scale | 1.875       3.889
    WIMSE var. weight | 0.132       0.017
    WIMSE bias weight | 0.868       0.983

```

The first set of output indicates that 595 observations were to the left of the cutoff or threshold of `margin` = 0, that 702 were to the right, and that the two curves are fourth-order polynomials that use the entire range of the running variable. The second set of output indicates that 15 equal-spaced bins were selected below the cutoff and 35 above. From the top of the output, the number of bins was selected using the mimicking variance method, and the bin means were computed using spacing estimators. The third set of output indicates that minimizing integrated MSE would lead to considerably fewer bins.

The two fitted fourth-degree polynomial curves given in the second panel of figure [25.5](#) show a clear discontinuity at `margin` = 0, suggesting that there is an incumbency advantage. The bin means are consistent with this. Both the curves and binned means suggest an increasing relationship between `vote` and `margin`, aside from the five lowest bins.

The global polynomials lead to the following estimate of the TE.

```

. * SRD: ATE estimate using parametric fourth-order polynomials
. generate d = margin > 0
. qui regress vote i.d##c.margin##c.margin##c.margin##c.margin, vce(robust)
. di "Parametric ATE = " %6.3f _b[1.d] " het-robust st. error = " %6.3f _se[1.d]
Parametric ATE = 9.407 het-robust st. error = 1.659
. qui regress vote i.d##c.margin##c.margin##c.margin##c.margin, vce(cluster state)
. di "Parametric ATE = " %6.3f _b[1.d] " clu-robust st. error = " %6.3f _se[1.d]
Parametric ATE = 9.407 clu-robust st. error = 1.749

```

At the threshold of `margin` = 0, incumbency leads to an increase in the vote share in the subsequent election of 9.41 percentage points. This is a very large effect and is quite precisely estimated. The heteroskedastic–robust are similar to the cluster–robust standard errors.

The limitation of this estimate of the TE is that it relies on correct functional form specification, and because all the data are used, it will be influenced by observations a long way from those in the neighborhood of `margin` = 0.

25.7.5 The `rdrobust` command

The community-contributed `rdrobust` command provides an estimate of ATE using only observations in the neighborhood of the cutoff value of the running variable. This estimate is a local-polynomial estimate using only data in the region of the cutoff and will vary with polynomial degree, kernel, and bandwidth. Furthermore, local polynomial estimators are biased, so a bias correction may be warranted. The command sets defaults for these various factors but allows the user great flexibility in departing from the defaults.

The general syntax for the `rdrobust` command is

```
rdrobust depvar runvar [ if ] [ in ] [ , options ]
```

Table 25.3. Selected options of `rdrobust` command

options	Description
<code>c(cutoff)</code>	the RD cutoff in the running variable
<code>p(pvalue)</code>	order of local polynomial for estimation (default <code>p(1)</code>)
<code>q(qvalue)</code>	order of local polynomial for bias correction (default <code>q(2)</code>)
<code>kernel(kernelfn)</code>	kernel function used (default <code>kernel(triangular)</code>)
<code>bwselect(bwmethod)</code>	bandwidth selection procedure (default <code>bwselect(mserd)</code>)
<code>vce(vcemethod)</code>	variance-covariance matrix estimator (default <code>vce(nn 3)</code>)
<code>all</code>	report three different <code>ate</code> and <code>vce</code> estimators
<code>cov(covars)</code>	additional covariates for estimation and inference
<code>fuzzy(fuzzyvar)</code>	the treatment variable used in fuzzy regression
<code>deriv(dvalue)</code>	discontinuity (FRD) estimation set <code>dvalue</code> to 1 for kink RD estimation

Table 25.3 provides a partial list of the comprehensive options. Heteroskedastic-robust and cluster-robust standard errors can be obtained based on either nearest-neighbor methods or residual-based standard errors—see [Calonico et al. \(2017\)](#) and articles cited therein for details. The `cov()` option is unnecessary, but the inclusion of pretreatment variables as additional covariates can improve estimator precision. For the moment, we consider only SRD and defer discussion of the `fuzzy()` and `deriv()` options.

Using command defaults, aside from adding the `all` option, we obtain

```
. * SRD: rdrobust command using command defaults aside from option all
. rdrobust vote margin, c(0) all
```

Sharp RD estimates using local polynomial regression.

Cutoff c = 0	Left of c	Right of c	Number of obs =	1297
Number of obs	595	702	BW type =	mserd
Eff. Number of obs	359	322	Kernel =	Triangular
Order est. (p)	1	1	VCE method =	NN
Order bias (q)	2	2		
BW est. (h)	17.708	17.708		
BW bias (b)	27.984	27.984		
rho (h/b)	0.633	0.633		

Outcome: `vote`. Running variable: `margin`.

Method	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
Conventional	7.416	1.4604	5.0782	0.000	4.55378 10.2783
Bias-corrected	7.5099	1.4604	5.1425	0.000	4.64768 10.3722
Robust	7.5099	1.7426	4.3095	0.000	4.09441 10.9255

The command defaults use local linear regressions with a triangular kernel, the method generally favored by the literature. The key choice then is the bandwidth; see section [25.7.6](#) for a summary of the various options. The default `bwselect(mserd)` option leads to a large bandwidth so that observations with `margin` in the range $(-17.708, 17.708)$ are included; this uses around one-half of the sample, so the efficiency loss of a local analysis should not be great.

The reported conventional estimate of 7.416 equals $(y^+ - y^-)$, where, for example, y^+ is the prediction at `margin = 0` from local linear regression using observations with `margin` in the range $(0, 17.708)$. The associated standard error of 1.4604 is computed using a heteroskedastic-robust estimate that by default is based on nearest-neighbor matching of residuals; the default uses three nearest neighbors. Alternative options use residual-based White standard errors, possibly with small-sample adjustment. The residual-based methods require a choice of bandwidth. The bandwidth used is the same as that chosen for estimation, a bandwidth that is not optimal for variance estimation and may lead to poor finite sample performance. Cluster-robust versions of these methods are also available. Recall that a parametric fourth-order global polynomial led to estimate 9.407 and standard error 1.659.

Nonparametric estimators such as local polynomial are biased because the bandwidth is chosen to minimize MSE, which involves a tradeoff between bias and variance. The reported bias-corrected estimate of 7.510 involves a recentering based on an estimate of the bias that by default uses a second-order local polynomial and a bandwidth of 27.984 in this application. The 95% confidence interval is then $7.5099 \pm 1.960 \times 1.4604$.

The bias-corrected estimate can perform poorly in finite samples because of noise in estimation of the bias. The reported robust estimate given in the final line of the output provides a larger standard error of 1.7426 that accounts for this additional source of estimation error.

The preceding local linear estimate of the ATE can be represented graphically by using the `rdplot` command with bandwidths specified to equal those selected by the `rdrobust` command, bandwidths that are stored in `e(h_l)` and `e(h_r)`. We have

```

. * SRD: rdplot of the local linear estimate of ATE obtained by rdrobust
. qui rdrobust vote margin, c(0)
. qui rdplot vote margin if -e(h_l)<= margin & margin <= e(h_r),
>     binselect(esmv) kernel(triangular) h(`e(h_l)` `e(h_r)` ) p(1)
>     graph_options(title("RD plot of the default ATE estimate")
>                 ytitle("Vote share at t+6") xtitle("Margin of victory at t"))

```

The first panel of figure 25.6 shows the two local linear lines fit for observations with `margin` in the ranges $(-17.708, 0)$ and $(0, 17.708)$. The estimated TE of 7.416 is the vertical distance between the two fitted lines at `margin = 0`.

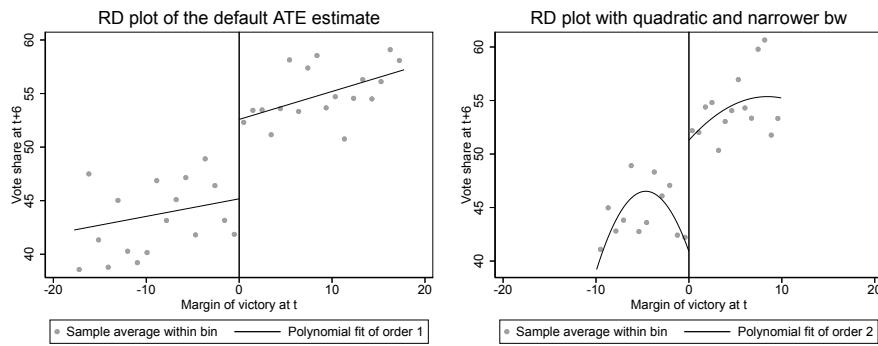


Figure 25.6. SRD: ATE estimate using local linear and local quadratic regression

As a cautionary tale regarding overfitting, the second panel of figure 25.6 presents an alternative estimate that uses a narrower bandwidth and a local quadratic estimate.

```

. * SRD: rdplot of the local quadratic estimate with narrower bandwidth
. qui rdplot vote margin if -10 <= margin & margin <= 10,
>     binselect(esmv) kernel(uniform) h(`-5` `5` ) p(2)
>     graph_options(title("RD plot with quadratic and narrower bw")
>                 ytitle("Vote share at t+6") xtitle("Margin of victory at t"))

```

For any given data application, the TE estimate will vary according to many estimator choices. There is general agreement that it is best to use local linear regression. The bandwidth choice then becomes crucial, and at a minimum, one should check the robustness of the estimate to variation in the bandwidth.

The following illustrates how results change with departures from a reference estimate of local linear with a triangular kernel, bandwidths of 10 on

either side of the cutoff, and standard errors based on nearest-neighbor matching of residuals.

```
. * SRD: rdrobust estimates with different estimation settings
. qui rdrobust vote margin, c(0) all h(10 10) rho(0.5) p(1) kernel(triangular)
. estimates store P1tri
. qui rdrobust vote margin, c(0) all h(10 10) rho(0.5) p(1) kernel(tri) vce(hc0)
. estimates store sewhite
. qui rdrobust vote margin, c(0) all h(10 10) rho(0.5) p(1) kernel(uniform)
. estimates store P1unif
. qui rdrobust vote margin, c(0) all h(15 15) rho(0.5) p(2) kernel(triangular)
. estimates store P2tri
. qui rdrobust vote margin, c(0) all h(5 5) rho(0.5) p(2) kernel(triangular)
. estimates store narrow
. estimates table P1tri sewhite P1unif P2tri wide narrow, b(%6.3f) se
> stfmt(%6.0f) stats(N_h_l N_h_r )
```

Variable	P1tri	sewhite	P1unif	P2tri	wide	narrow
Conventional	7.985 1.838	7.985 1.831	6.899 1.722	11.922 2.718	9.086 2.241	13.133 3.594
Bias-corrected	11.922 1.838	11.922 1.831	10.390 1.722	14.896 2.718	12.659 2.241	10.619 3.594
Robust	11.922 2.718	11.922 2.660	10.390 2.670	14.896 3.406	12.659 2.939	10.619 5.363
N_h_l	245	245	245	245	319	128
N_h_r	206	206	206	206	288	117

Legend: b/se

We focus on the conventional estimates. White standard errors are quite close to the nearest-neighbor standard errors. Changing to a uniform kernel reduces the TE from 7.985 to 6.899. The local quadratic estimate of 11.922 is much larger than the local linear and has a much larger standard error. A wider bandwidth leads to a larger estimate of 9.086 and a larger standard error, despite use of a greater fraction of the sample, indicating that the linear model does not fit as well over a wider range. A narrower bandwidth also leads to a larger estimate and a larger standard error.

25.7.6 The rdbwselect command

The community-contributed `rdbwselect` command selects the bandwidth used to estimate the RD TE. It is used within `rdrobust` and is also a stand-alone command.

The command uses two general approaches to bandwidth selection, both of which are plugin methods rather than cross-validation. The first is a plugin formula that minimizes MSE and entails estimation of bias and variance. The second approach provides optimal coverage rates for confidence intervals and entails a rescaling of the first plugin formula. For details on these methods, see [Calonico et al. \(2017\)](#). Within each method, the bandwidth may be the same or may differ on either side of the cutoff, and the bandwidth may be chosen to be optimal for the sum of the regression estimates rather than the difference (the TE).

The following example displays all available bandwidth selection methods.

```
. * SRD: rdbwselect command showing all available bandwidth selection methods
. rdbwselect vote margin, c(0) all
```

Bandwidth estimators for sharp RD local polynomial regression.

Cutoff c = 0	Left of c	Right of c	Number of obs =	1297
Number of obs	595	702	Kernel	= Triangular
Min of margin	-100.000	0.036	VCE method	= NN
Max of margin	-0.079	100.000		
Order est. (p)	1	1		
Order bias (q)	2	2		

Outcome: vote. Running variable: margin.

Method	BW est. (h)		BW bias (b)	
	Left of c	Right of c	Left of c	Right of c
mserd	17.708	17.708	27.984	27.984
msetwo	16.154	18.009	27.096	29.205
msesum	18.326	18.326	31.280	31.280
msecomb1	17.708	17.708	27.984	27.984
msecomb2	17.708	18.009	27.984	29.205
<hr/>				
cerrd	12.374	12.374	27.984	27.984
certwo	11.288	12.585	27.096	29.205
cersum	12.806	12.806	31.280	31.280
cercomb1	12.374	12.374	27.984	27.984
cercomb2	12.374	12.585	27.984	29.205

The various methods lead to bandwidths similar to those selected by the default `mserd` method. The biggest difference is that the `cer` methods lead to a smaller bandwidth below the cutoff.

25.7.7 Fuzzy regression discontinuity design

The SRD design assumes that on one side of the cutoff, all observations are untreated, while on the other side of the cutoff, all observations are treated. FRD design relaxes this sharp cutoff. If those above the cutoff are on average more likely to be treated, for example, then under FRD some of the individuals below the cutoff may be untreated or some above the cutoff may not be treated, or both.

Consider the local estimation approach. Intuitively, the SRD TE $y^+ - y^-$ needs to be scaled up by dividing by the fraction of the sample who move from untreated to treated in the neighborhood of the cutoff. This leads to the FRD TE

$$\text{ATE}_{\text{FRD}} = \frac{y^+ - y^-}{D^+ - D^-}$$

where D is a binary indicator of actual treatment status,
 $D^+ = \lim_{x \rightarrow c^+} E(D_i|x_i = c)$ and $D^- = \lim_{x \rightarrow c^-} E(D_i|x_i = c)$.

The LATE framework of section [25.5](#) for binary treatment D and a binary instrument z is relevant. For FRD, the running variable x is essentially forming a binary instrument z that switches from 0 to 1 (or from 1 to 0) on either side of the threshold. As for LATE, we potentially have compliers, defiers, always-takers, and never-takers. We make the monotonicity assumption that $D_i(x)$ is nonincreasing (or nondecreasing) in x at $x = c$ for all i when treatment is more likely above (or below) the cutoff. This rules out defiers so that $D^+ - D^-$ measures compliers.

It follows that FRD measures the TE for compliers. As for LATE, this restrictive interpretation limits the possibility of extrapolating the conclusion beyond the complier subpopulation.

The local FRD estimate can be obtained as the ratio of an estimate of the numerator, obtained by local polynomial estimation of y on x on either side of the threshold, to an estimate of the denominator, obtained by local polynomial estimation of D on x on either side of the threshold. These two components

can be computed using the separate commands `rdrobust y x` and `rdrobust D x`.

It is more convenient to use the `fuzzy()` option of the `rdrobust` command. The basic command is `rdrobust y x, fuzzy(D)`, where `D` is the treatment indicator. This provides standard errors of the estimate and by default follows the recommended procedure of selecting the optimal bandwidth for the outcome local polynomial regression and using the same bandwidth for the treatment local polynomial regression.

As illustration, we continue with the same dataset but change the treatment status. In SRD, no observations with `margin < 0` are treated, and all observations with `margin >= 0` are treated. Instead, we suppose 10% of observations with `margin < 0` do get treated, and only 60% of observations with `margin >= 0` are treated. We have

```
. * FRD DGP: 10% below cutoff are treated and 40% above cutoff are treated  
. gen dtreat = margin > 0  
. set seed 10101  
. qui replace dtreat = 1 if (runiform() < 0.1 & margin < 0)  
. qui replace dtreat = 0 if (runiform() < 0.4 & margin > 0)
```

The `rdplot` command can be used to plot the treatment indicator against the running variable using binned data and a fitted fourth-order polynomial curve.

```
. * FRD: rdplot command for treatment indicator against margin  
. qui rdplot dtreat margin, c(0)  
> graph_options(title("Fuzzy rd: rdplot of treatment on running variable"))  
> ytitle("Treatment indicator at t") xtitle("Margin of victory at t"))
```

Figure 25.7 shows that around 10% of those below the cutoff were treated, and around 60% above the cutoff were treated. So one-half of the sample was compliers, switching from no treatment below cutoff to treatment above the cutoff. We earlier estimated $y^+ - y^-$ to equal 7.985, so we expect the FRD estimate for compliers to be approximately $7.985/0.5 = 15.97$.

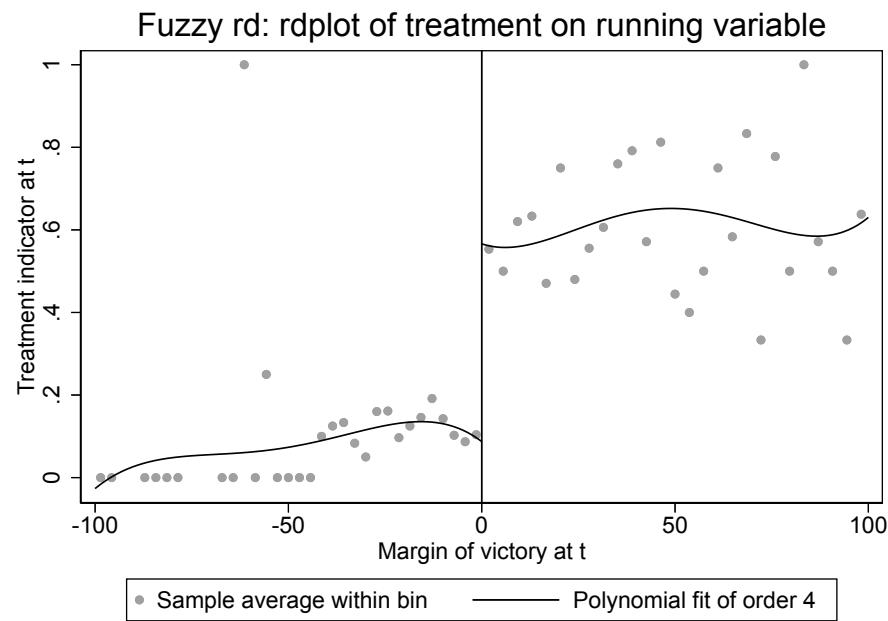


Figure 25.7. SRD: ATE estimate using local linear and local quadratic regression

Using the `fuzzy()` option of `rdrobust` yields

```
. * Fuzzy RD: rdrobust with fuzzy() option
. rdrobust vote margin, c(0) fuzzy(dtreat) all
Fuzzy RD estimates using local polynomial regression.
```

Cutoff c = 0	Left of c	Right of c	Number of obs = 1297
Number of obs	595	702	BW type = mserd
Eff. Number of obs	380	338	Kernel = Triangular
Order est. (p)	1	1	VCE method = NN
Order bias (q)	2	2	
BW est. (h)	18.831	18.831	
BW bias (b)	32.219	32.219	
rho (h/b)	0.584	0.584	

First-stage estimates. Outcome: dtreat. Running variable: margin.

Method	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
Conventional	.4475	.06318	7.0831	0.000	.323673 .571329
Bias-corrected	.44613	.06318	7.0613	0.000	.322298 .569954
Robust	.44613	.07393	6.0346	0.000	.301229 .591023

Treatment effect estimates. Outcome: vote. Running variable: margin. Treatment Sta > tus: dtreat.

Method	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
Conventional	16.405	4.0324	4.0682	0.000	8.50133 24.308
Bias-corrected	16.569	4.0324	4.1091	0.000	8.66616 24.4728
Robust	16.569	4.7527	3.4863	0.000	7.25428 25.8847

A first-order local polynomial is fit with the same bandwidth for the outcome regression and the treatment regression. The first-stage conventional estimate of 0.4475 is an estimate of $D^+ - D^-$. The command does not provide output on $y^+ - y^-$, but it is 7.3412, slightly different from the earlier SRD estimate because the bandwidth is slightly different. This leads to the treatment conventional estimate of $7.3412/0.4475 = 16.405$. The TE for compliers is 16.405.

25.7.8 Further discussion

This application has applied a pooled regression to a state panel dataset. In many panel applications, fixed effects are used to identify a causal effect. In an RD design, however, it is unnecessary for identification to include unit-specific fixed effects.

Also, with a state-year panel, it is standard to obtain cluster-robust standard errors, with clustering on state. Here the cutoff indicator variable D has little

within-state correlation, and adding option `vce(cluster state)` to the `rdrobust` command increases standard errors by around 5%.

A useful check of the RD design is to perform placebo tests that replace the outcome variable with other variables that should not show a jump in the running variable at the cutoff.

Here we do so using the population in the state and using the Senate vote in an earlier election. Senate elections are every two years on a rotating 6-year cycle, and variable `vote` is measured 6 years after variable `margin`; so to compute the vote six years before variable `margin`, we need to lag variable `vote` by 12 years or 6 2-year periods. We have

```
. * SRD: Placebo test
. qui rdplot population margin, c(0)
> graph_options(title("Placebo rdplot using population")
>                 ytitle("Population at t + 6") xtitle("Margin of victory at t"))
. tsset state year
Panel variable: state (unbalanced)
Time variable: year, 1914 to 2010, but with gaps
    Delta: 1 unit
. generate votelagged = 0
. replace votelagged = 16.vote // Six two-year periods ago = 12 years
(1,377 real changes made, 256 to missing)
. qui rdplot votelagged margin, c(0)
> graph_options(title("Placebo rdplot using lagged vote")
>                 ytitle("Vote share at t - 6") xtitle("Margin of victory at t"))
```

Figure 25.8 shows no jump at `margin > 0`, and the curves have the expected shape with little relationship between population and `margin` and an increasing relationship between the lagged vote and future margin of victory.

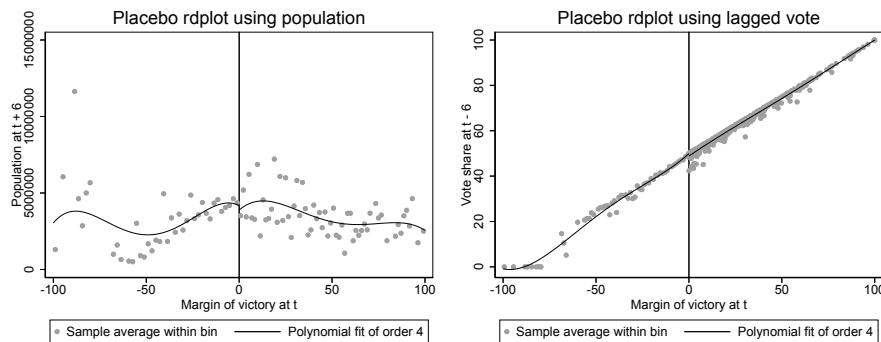


Figure 25.8. SRD: Placebo plots

The SRD and FRD designs consider a vertical jump in the running variable at the cutoff. An extension, called kink RD design, instead allows for a change in slope in the relationship between the outcome and the running variable. For details see [Card et al. \(2015\)](#). The `deriv(1)` option of the `rdrobust` command estimates the TE in a kink RD design (up to scale), and the additional `fuzzy()` option estimates a fuzzy kink RD design. Identification relies on L'Hopital's rule and no jump in the first stage, and the data demands for kink RD are much greater than those for SRD.

25.8 Conditional quantile regression with endogenous regressors

A major attraction of quantile regression (QR), introduced in chapter 15, is that it permits responses to changes in a key policy variable to differ across individuals. For example, a training program may have greater effect for individuals with lower conditional quantile of earnings.

In many such applications, the policy variable is endogenous. For example, individuals may self-select into the training program. The extension from linear model IV to quantile IV is challenging. Stronger assumptions are needed, different proposed methods make different assumptions, some methods are computationally difficult to implement, and this is still an active area of research. Some leading methods are based on the TES methods for binary treatments. The discussion here is very brief; see the original articles for complete details.

25.8.1 Local conditional quantile TE with endogenous binary regressor

We consider estimation of the quantile treatment effect (QTE) with D that is binary and an instrument that is also binary.

Let y denote a continuous outcome, \mathbf{x} denote control variables, D denote a binary treatment that takes value 0 or 1, and z denote a binary instrument that takes value 0 or 1. Interest lies in estimating the conditional QTE

$$\Delta_q(\mathbf{x}) = Q_q(y|D=1, \mathbf{x}) - Q_q(y|D=0, \mathbf{x})$$

If D and \mathbf{x} are exogenous, then this is easily estimated as the coefficient of D following conditional quantile regression (CQR) of y on \mathbf{x} and D that minimizes

$$Q(\boldsymbol{\beta}_q, \delta_q) = \sum_{i=1}^N \rho_q(y_i - D_i \delta_q - \mathbf{x}'_i \boldsymbol{\beta})$$

where $\rho_q(\cdot)$ is the check function $\rho_q(u) = u\{q - \mathbf{1}(u < 0)\}$ introduced in section 15.2.1.

Instead, we consider a binary treatment D that is endogenous. [Abadie, Angrist, and Imbens \(2002\)](#) proposed a local average QTE estimator that is an extension to quantile regression of the LATE estimator of [Abadie \(2003\)](#) for linear regression presented in section [25.5.3](#).

For simplicity, consider the case that an increase in the instrument pushes individuals toward treatment. Ideally, treatment $D = 1$ when $z = 1$; such individuals are called compliers. It is possible that instead $D = 0$ when $z = 1$; such people are called defiers. The following analysis assumes that there are no defiers (individuals for whom $D = 0$ when $z = 1$) after additionally conditioning on \mathbf{x} ; an assumption also called the monotonicity assumption.

The binary instrument z is assumed to satisfy four conditions, all of which apply after conditioning on \mathbf{x} , that we briefly summarize as 1) z is independent of y and D ; 2) z has no direct effect on y ; 3) z is a relevant instrument; 4) there are no defiers.

The method of [Abadie, Angrist, and Imbens \(2002\)](#) controls for the endogeneity of D by using a weighted version of CQR. The local average conditional QTE at the q th quantile, Δ_q , is estimated by $\widehat{\delta}_q$, where $\widehat{\delta}_q$ and $\widehat{\beta}_q$ minimize the weighted sum

$$S(\boldsymbol{\beta}_q, \delta_q) = \sum_{i=1}^N \widehat{\kappa}_i \rho_q(y_i - D_i \delta_q - \mathbf{x}'_i \boldsymbol{\beta}_q)$$

where the weights $\widehat{\kappa}_i$ are consistent estimates of the same kappa weights κ_i as defined in [\(25.9\)](#). Note that this reduces to the usual CQR estimator defined in chapter 15 if $\kappa_i = 1$ for all i .

25.8.2 The ivqte command

The community-contributed `ivqte` command, due to [Frölich and Melly \(2010\)](#), implements the preceding estimator as well as estimators for unconditional QTEs that are presented in the subsequent section. Running this command requires prior installment of the community-contributed `moremata` and `kdens` packages.

The command is restricted to a binary treatment and binary instrument; a working paper version of [Frölich and Melly \(2013\)](#) discusses adaptation to nonbinary instrument or to multiple instruments. The syntax for the command varies with whether treatment is endogenous or exogenous and whether a conditional or unconditional QTE is desired. The default is to not compute the variance of estimated coefficients. The `variance` option computes the variance using an analytical formula that allows for heteroskedasticity. This variance estimate uses kernel methods, and it may be necessary to change some related settings from their default values. Alternatively, a bootstrap can be used.

We revisit the CQR application of section 15.3 that uses data from the Medical Expenditure Panel Survey. The dependent variable `ltotexp` is the log of total medical expenditure by the Medicare elderly. The explanatory variables are an indicator for supplementary private insurance (`suppins`), one health-status variable (`totchr`), and three sociodemographic variables (`age`, `female`, `white`). We consider estimation at the median and use the `ivqte` command; the same estimates are obtained using `qreg`. The syntax of the `ivqte` command requires that a treatment variable be explicitly identified by being placed in parentheses and that it appear last. We initially treat all regressors as exogenous and obtain

```

. * Conditional QTE of exogenous suppins using ivqte (same as qreg)
. qui use mus203mepsmedexp, clear
. drop if ltotexp == .
(109 observations deleted)
. ivqte ltotexp totchr age female white (suppins), quantile(0.5) variance
Quantile regression
Estimator suggested in Koenker and Bassett (1978)
Quantile: .5
Dependent variable: ltotexp
Regressor(s): suppins totchr age female white
Number of observations: 2955

```

ltotexp	Coefficient	Std. err.	z	P> z	[95% conf. interval]
suppins	.2769771	.0572835	4.84	0.000	.1647034 .3892508
totchr	.3942664	.0215262	18.32	0.000	.3520758 .436457
age	.0148666	.0043557	3.41	0.001	.0063296 .0234036
female	-.0880967	.0564343	-1.56	0.119	-.1987058 .0225124
white	.4987457	.2051237	2.43	0.015	.0967107 .9007807
_cons	5.648891	.3825663	14.77	0.000	4.899075 6.398707

This standard conditional median regression yields a conditional QTE at the median of 0.277.

We next treat `suppins` as endogenous, with binary instrument `marry`, which is an indicator of marital status; about 56% of the sample are married. This example is illustrative; it makes the questionable assumption that `marry` can be excluded from the model for `ltotexp`.

The local average conditional QTE can be obtained using the `aai` option of the `ivqte` command. A local logit kernel regression (see section [17.8.4](#)) is used to obtain $\hat{p}(z_i)$, which appears in the weights defined in [\(25.9\)](#). The kernel varies with the type of regressor, so the syntax of the `ivqte` command requires distinction between continuous, binary, and ordered discrete regressors. We use the default choices of `kernel()`, `bandwidth()` for continuous regressors, and `lambda()` for binary regressors. The default is to restrict the predicted probabilities $\hat{p}(z_i)$ to the interval (0.001, 0.999), which can lead to weights as large as 1,000 being given to some observations. The `trim()` option changes this default value.

We obtain

```

. * Conditional QTE of endogenous suppins using ivqte with aai option
. ivqte ltotexp (suppins=marry), dummy(white female) continuous(totchr age)
> quantile(0.5) var trim(0.01) generate_p(predprob) aai
IV quantile regression
Estimator suggested in Abadie, Angrist and Imbens (2002)

Quantile(s): .5
Dependent variable: ltotexp
Treatment variable: suppins
Instrumental variable: marry
Control variable(s): totchr age white female
Number of observations: 2955
Proportion of compliers: .063

Propensity score estimated by local logit regression with h = infinity and
> lambda = 1
Positive weights estimated by local linear regression with h = infinity and
> lambda = 1
Variance estimated using local linear regression with h = infinity and lambda = 1

```

	Coefficient	Std. err.	z	P> z	[95% conf. interval]
suppins	1.179383	.5203996	2.27	0.023	.1594189 2.199348
totchr	.4590747	.1834216	2.50	0.012	.0995749 .8185745
age	-.0087295	.0572944	-0.15	0.879	-.1210245 .1035655
white	.2838135	1.349665	0.21	0.833	-2.361482 2.929109
female	-.3344497	1.129468	-0.30	0.767	-2.548167 1.879267
_cons	7.616818	5.273967	1.44	0.149	-2.719968 17.9536

The conditional local QTE estimate is 1.179, exceptionally large and more than four times larger than the estimate assuming exogeneity of `suppins`. The associated standard error is nine times larger, increasing to 0.520, in part because the correlation between `suppins` and the instrument `marry` is low. Note also that the proportion of compliers is only 0.063, so this illustrative example is not well suited to this method. The predicted probabilities $\hat{p}(z_i)$, saved in the variable named `predprob`, range from 0.141 to 0.918, so in this example, there was no need to trim.

From additional regressions, the conditional local QTE estimate is 1.374 at $q = 0.25$ and 1.032 at $q = 0.75$. These local QTE estimates apply to compliers.

25.8.3 Alternative estimators of conditional QTE with endogeneity

The preceding model is quite restrictive. For example, the effect of treatment might be interactive with some regressors, in which case multiple

instruments would be needed. Several other estimators of the conditional QTE have been proposed. These are based on alternative models and assumptions and are not yet widely used, in part because of computational challenges.

Consider linear CQR with regressors \mathbf{d} and \mathbf{x} . At the q th quantile, the parameters minimize $S(\boldsymbol{\beta}_q, \boldsymbol{\delta}_q) = \sum_{i=1}^N \rho_q(y_i - \mathbf{d}'_i \boldsymbol{\delta}_q - \mathbf{x}'_i \boldsymbol{\beta}_q)$. When \mathbf{d} is instead endogenous and instruments \mathbf{z} are available, [Chernozhukov and Hansen \(2008\)](#) provide theory that leads to the following objective function,

$$S(\boldsymbol{\beta}_q, \boldsymbol{\delta}_q, \boldsymbol{\gamma}_q) = \sum_{i=1}^N w_i \rho_q \left(y_i - \mathbf{d}'_i \boldsymbol{\delta}_q - \mathbf{x}'_i \boldsymbol{\beta}_q - \hat{\mathbf{d}}'_i \boldsymbol{\gamma}_q \right)$$

where in the simplest case $w_i = 1$ for all i and the additional variables $\hat{\mathbf{d}}_i$ are obtained by least-squares projection of \mathbf{d}_i on \mathbf{z}_i and \mathbf{x}_i ; even more simply, one might use \mathbf{z}_i .

The authors propose obtaining an estimate of $\boldsymbol{\delta}_q$ as follows. First, for each of a range of values of $\boldsymbol{\delta}_q$, run the ordinary CQR to obtain estimates $\hat{\boldsymbol{\beta}}_q(\boldsymbol{\delta}_q)$ and $\hat{\boldsymbol{\gamma}}_q(\boldsymbol{\delta}_q)$. Then, find that value of $\boldsymbol{\delta}_q$ that makes the coefficient of the added variables $\hat{\mathbf{d}}_i$ as small as possible by minimizing a quadratic form in $\hat{\boldsymbol{\gamma}}_q(\boldsymbol{\delta}_q)$, where the weighting matrix is the inverse of asymptotic variance of $\hat{\boldsymbol{\gamma}}_q(\boldsymbol{\delta}_q)$. The method permits overidentified models and provides weak identification robust inference.

This estimator, often called IV QR, relies in part on the assumption of rank similarity, detailed below in section [25.8.4](#). It extends the original work of [Chernozhukov and Hansen \(2006\)](#). Recent work by [Franguridi, Gafarov, and Wüthrich \(2020\)](#) provides finite sample theory.

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 $Q_q(y^* | \mathbf{x}) = \mathbf{x}' \boldsymbol{\beta}_q$.

[Chernozhukov et al. \(2019\)](#) propose an alternative method for CQR with a continuous endogenous regressor, one that can additionally allow for censoring; see section 15.3.11. The community-contributed `cqr` command ([Chernozhukov et al. 2019](#)) implements these methods.

[Machado and Santos Silva \(2019\)](#) provide yet another approach that restricts attention to the location-scale linear regression model

$$y_i = \mathbf{x}'_i \boldsymbol{\beta} + g(\mathbf{w}'_i \boldsymbol{\gamma}) \times u_i$$

where \mathbf{w}_i are transformations of \mathbf{x}_i and $g(\cdot) > 0$ is a specified function. Then conditional quantiles are linear in \mathbf{x} with

$$Q_q(y_i|\mathbf{x}_i) = \mathbf{x}'_i \{\boldsymbol{\beta} + Q_q(u_i|\mathbf{x}_i)\}$$

In this model, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, and the q th conditional quantile of the error given \mathbf{x} determine the q th conditional quantile of y given \mathbf{x} . The assumptions that u_i is independent and identically distributed with $E(u) = 0$ and the normalization that $E(|u|) = 1$ yield two moment conditions that enable estimation of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ by two OLS regressions. The quantile $Q_q(u_i|\mathbf{x}_i)$ can then be estimated from the scaled errors $(y_i - \mathbf{x}'_i \hat{\boldsymbol{\beta}})/g(\mathbf{w}'_i \hat{\boldsymbol{\gamma}})$.

[Machado and Santos Silva \(2019\)](#) extend this approach to the case where some of the regressors are endogenous and instruments are available. Then the model is qualitatively similar to the model of [Chernozhukov and Hansen \(2008\)](#), being more flexible by allowing nonlinear quantile effects but more restrictive because [Chernozhukov and Hansen \(2008\)](#) essentially allow for random coefficients. Estimation is much simpler than the method proposed by [Chernozhukov and Hansen \(2008\)](#) and can be performed using the community-contributed program `ivqreg2`, due to [Machado and Santos Silva \(2019\)](#).

25.8.4 Rank invariance and rank similarity

Interpreting QTE when treatment is not randomly assigned requires additional assumptions. The local conditional QTE relies on the assumption of monotonicity of treatment choice. Other methods can require rank invariance or rank similarity.

For a binary treatment, let Y_1 and Y_0 denote the potential outcomes under, respectively, treatment or no treatment; for a given individual, we observe only one of Y_1 or Y_0 . The rank of an individual is given by the cumulative distribution function (c.d.f.), which for a continuous random variable takes values that are uniformly distributed on $(0, 1)$. Define the potential rank if treated as $U_1 = F_1(Y_1)$ and the potential rank if not treated as $U_0 = F_0(Y_0)$. Then $U_1 \sim U(0, 1)$ and $U_0 \sim U(0, 1)$.

Let potential outcomes depend on observables \mathbf{x} and an unobservable scalar v , so $Y_1 = g_1(\mathbf{x}, v)$ and $Y_0 = g_0(\mathbf{x}, v)$. Then the potential ranks are $U_1 = F_1\{g_1(\mathbf{x}, v)\}$ and $U_0 = F_0\{g_0(\mathbf{x}, v)\}$.

Under rank invariance, also called rank preservation, after one controls for observables and a scalar unobservable, a person's potential rank is the same with treatment or without treatment. Thus, $U_1|(\mathbf{x}, v) = U_0|(\mathbf{x}, v)$ for all (\mathbf{x}, v) . This in turn implies that, conditioning on the observables \mathbf{x} alone, $U_1|\mathbf{x}$ and $U_0|\mathbf{x}$ have identical distribution for all \mathbf{x} .

Rank similarity weakens this last result to say that, conditioning on both observables \mathbf{x} and the unobservable v , $U_1|(\mathbf{x}, v)$ and $U_0|(\mathbf{x}, v)$ have identical distribution for all (\mathbf{x}, v) . This allows for random departures from rank invariance, qualitatively similar to a student with given ability being ranked differently from test to test, due purely to randomness in performance from test to test.

[Melly and Wüthrich \(2017\)](#) provide a discussion of the role in QR of assumptions of rank invariance and rank invariance and also monotonicity assumptions. [Dong and Shen \(2018\)](#) present nonparametric tests for rank invariance and for rank similarity.

25.9 Unconditional quantiles

An important distinction is that between the ME of changing a regressor on features of the conditional distribution of $y|x$ and the ME of changing a regressor on features of the unconditional distribution of y . The presentation of QR in chapter 15 and section [25.8](#) so far has considered the former case, but the latter case can also be of great interest for policy makers.

The literature on unconditional quantiles has focused on the TES case, where interest lies in the ME of changing a binary treatment. For example, interest may lie in the effect of a training program on the 25th percentile of earnings, rather than on the 25th percentile of earnings conditional on control variables. This is called the unconditional QTE.

For linear regression, the conditional and unconditional MES are the same, equaling β . For quantile regression, there is a difference. As pointed out by [Frölich and Melly \(2010\)](#),

The interpretation of the unconditional effects is slightly different from the interpretation of the conditional effects, even if the conditional QTE is independent from the value of X . This is because of the definition of the quantile. For instance, if we are interested in a low quantile, the conditional QTE will summarize the effect for individuals with relatively low Y even if their absolute level of Y is high. The unconditional QTE, on the other hand, will summarize the effect with a relatively low absolute Y .

[Borah and Basu \(2013\)](#) discuss in detail the distinction between conditional and unconditional quantile regression.

In this section, we present several methods. The first specializes to the case of binary treatment D and computes the unconditional QTE as the difference in weighted quantile estimates according to whether $D = 1$ or $D = 0$, where the weights are determined by the probability of treatment given regressors x . The second approach uses influence functions that consider only small changes in the treatment. The third approach estimates semiparametric estimates of conditional c.d.f.'s or conditional quantiles and

then integrates over regressors \mathbf{x} . The section concludes by extending the first approach to allow treatment D to be endogenous.

In the current example, the explanatory power of the regressors is low, with an R^2 from OLS regression of 0.19. So we might expect unconditional QTES to be not too different from conditional QTES; both depend greatly on variation due to unobservables.

25.9.1 Unconditional QTE using inverse-propensity score weighting

[Firpo \(2007\)](#) estimates the unconditional QTE when the effects of a binary treatment vary across individuals by using inverse-propensity score weighting, a standard method used in the TES literature; see chapter [24](#). It is assumed that regressors control for any selection effects and that the treatment is rank preserving, so that if an individual is in the q th quantile without treatment, he or she is also in the q th quantile with treatment. The estimation method actually does not require QR regression.

We wish to estimate $\Delta_q = Q_q(y|D=1) - Q_q(y|D=0)$, controlling for regressors \mathbf{x} . Let $\hat{p}(\mathbf{x}_i)$ be an estimate of $\Pr(D_i = 1|\mathbf{x}_i)$, the probability of treatment. For a treated observation, we weight by the inverse of the predicted probability of treatment, while for a nontreated individual, we weight by the inverse of the probability of nontreatment.

Specifically, from section 15.1, the q th raw quantile α_q minimizes $\sum_{i=1}^N \rho_q(y_i - \alpha_q)$, where $\rho_q(\cdot)$ is the check function. For observations with $D_i = 1$, we estimate $Q_q(y|D=1)$ by $\hat{\alpha}_{1q}$, which minimizes the weighted sum $\sum_{i=1}^N w_{1i}\rho_q(y_i - \alpha_{1q})$, where $w_{1i} = 1/\hat{p}(\mathbf{x}_i)$. For $Q_q(y|D=0)$, we use $\hat{\alpha}_{0q}$, which minimizes $\sum_{i=1}^N w_{0i}\rho_q(y_i - \alpha_{0q})$, where $w_{0i} = 1/\{1 - \hat{p}(\mathbf{x}_i)\}$.

Combining, we equivalently estimate Δ_q by $(\hat{\alpha}_{1q} - \hat{\alpha}_{0q})$ where $\hat{\alpha}_{1q}$ and $\hat{\alpha}_{0q}$ minimize

$$Q(\alpha_{1q}, \alpha_{0q}) = \sum_{i=1}^N w_i \rho_q\{y_i - D_i \alpha_{1q} - (1 - D_i) \alpha_{0q}\} \tag{25.11}$$

where the inverse-propensity score weights are

$$w_i = \frac{D_i}{\hat{p}(\mathbf{x}_i)} + \frac{1 - D_i}{1 - \hat{p}(\mathbf{x}_i)}$$

This estimator can be obtained using the `ivqte` command ([Frölich and Melly 2010](#)), introduced in section [25.8.2](#). The command syntax is similar to that presented for local conditional QTE estimation, except that the `aai` option is dropped and an instrument is not relevant. We obtain

```
. * Unconditional QTE of suppins on ltotexp using community-contributed
> command ivqte
. ivqte ltotexp (suppins), dummy(white female) continuous(totchr age)
> quantile(0.25(0.25)0.75) variance trim(0.1)
Unconditional Quantile Treatment Effects under exogeneity
Estimator suggested in Firpo (2007)
Quantile(s): .25 .5 .75
Dependent variable: ltotexp
Treatment variable: suppins
Control variable(s): totchr age white female
Number of observations: 2955
Propensity score estimated by local logit regression with h = infinity and lambda
> = 1
Variance estimated using local logit regression with h = infinity and lambda = 1
```

ltotexp	Coefficient	Std. err.	z	P> z	[95% conf. interval]
Quantile_1	.3989425	.074762	5.34	0.000	.2524117 .5454732
Quantile_2	.2826729	.0643581	4.39	0.000	.1565333 .4088124
Quantile_3	.2005053	.073077	2.74	0.006	.057277 .3437335

The unconditional QTES of `suppins` in this example are similar to the conditional QTES obtained by standard CQR using the `qreg` command; the biggest difference is at the upper quartile.

```

. * Corresponding conditional QTE of suppins on ltotexp using qreg command
. forvalues i = 1/3 {
2.     local j = `i'/4
3.     qui qreg ltotexp suppins totchr age white female, quant(`j') vce(robust)
4.     di "q = `j' "    b = " _b[suppins] "    se = " _se[suppins]
5. }
q = .25    b = .38583946    se = .05991632
q = .5     b = .27697708    se = .05346786
q = .75    b = .14885476    se = .06202515

```

25.9.2 Unconditional QTE using recentered influence functions

[Firpo, Fortin, and Lemieux \(2009\)](#) provide an estimator of the unconditional quantile effect that can be simply obtained by OLS regression with dependent variable recentered influence function (RIF) for the q th quantile, where the influence function measures the relative influence of individual observations on the value of a statistic.

Specifically, the RIF at y_q , the q th quantile of the dependent variable with value y , is

$$\text{RIF}(y; y_q) = y_q + \frac{q - \mathbf{1}(Y \leq y_q)}{f_Y(y_q)}$$

where $f_Y(y_q)$ is the density of Y at y_q .

An attraction of RIF regression is that this measure of the unconditional QTE can be used for continuous regressors, not just discrete regressors. The disadvantage is that it is akin to a derivative that considers only local changes in the regressors.

The RIF-based estimate can be implemented using the authors' `rifreg` command. Results depend crucially on the estimate of the density $f_Y(y_q)$. The `rifreg` command uses the `kdensity` command with default bandwidth the “optimal” width used by `kdensity` and with default kernel the Gaussian. Users should save and plot the kernel density estimates to see whether they appear reasonable.

We first obtain estimates for the 25th percentile of `ltotexp`, using the `generate()` option to save the kernel density estimates.

```
. * Unconditional QTE of suppins at median using community-contributed
> command rifreg
. rifreg ltotexp suppins totchr age female white, quantile(.25)
> generate(yval kdensval)
```

Source	SS	df	MS	Number of obs	=	2955
Model	1348.61919	5	269.723837	F(5, 2949)	=	115.81
Residual	8580.69514	2949	2.90969655	Prob > F	=	0.0000
Total	9929.31432	2954	3.36131155	R-squared	=	0.1358
				Adj R-squared	=	0.1344
				Root MSE	=	1.7058

rif_25	Coefficient	Robust				[95% conf. interval]
		std. err.	t	P> t		
suppins	.3996899	.065752	6.08	0.000	.2707653	.5286144
totchr	.4758505	.0218392	21.79	0.000	.4330288	.5186722
age	.0169063	.0050117	3.37	0.001	.0070796	.0267331
female	-.0100766	.0639374	-0.16	0.875	-.1354431	.11529
white	.5814719	.2226578	2.61	0.009	.1448915	1.018052
_cons	4.357429	.4249414	10.25	0.000	3.524217	5.190641

`rifreg` provides unconditional quantile estimates for all regressors. These are similar to the conditional estimates obtained from `qreg`, aside from `female`, which is highly statistically insignificant, and `white`. The command `scatter kdensval yval` gives a plot that suggests the kernel density estimates are reasonable.

Focusing on just the QTE for `suppins` at the quartiles, we obtain

```
. * Unconditional QTE of suppins at quartiles using community-contributed
> command rifreg
. forvalues i = 1/3 {
2. local j = `i'/4
3. qui rifreg ltotexp suppins totchr age white female, quantile(`j')
4. di "q = " `j' " b = " _b[suppins]
5. }
q = .25 b = .39968985
q = .5 b = .27289201
q = .75 b = .16171682
```

By comparison, from section [25.9.1](#) the method of [Firpo \(2007\)](#) yielded estimated QTEs of, respectively, 0.399, 0.283, and 0.201.

The RIF regression approach extends naturally to estimation of unconditional QTES in FE models because one can perform FE regression with dependent variable the RIF for the q th quantile. This can be implemented using the community-contributed command `xtrifreg` ([Borgen 2016](#)).

25.9.3 Counterfactual distributions

[Chernozhukov, Fernández-Val, and Melly \(2013\)](#) estimate conditional distributions and regressor distributions and combine these to obtain counterfactual distributions of many features, among other things, quantile effects, distribution effects, and Gini coefficients. The community-contributed `cdeco` and `counterfactual` commands (Chernozhukov, Fernández-Val, and Melly [2009](#)) implement these methods.

Key is obtaining an estimate of the c.d.f. of the outcome conditional on regressors. A parametric model for the c.d.f. can be used, but then results depend on very strong assumptions. The c.d.f. can be estimated nonparametrically (see [Li and Racine \[2007, chap. 6\]](#)), but there is then the curse of dimensionality.

Instead, the `method()` option of the `cdeco` command offers several other ways to estimate the conditional c.d.f. Specifying `qr` estimates many conditional quantiles that are then inverted to obtain the c.d.f. Specifying a `logit` (or `probit`) model fits a series of models for $\Pr(y > y_q | D, \mathbf{x})$ at various conditional quantiles y_q ; the default is $y_q = 0.01, 0.02, \dots, 0.99$. Specifying `cox` estimates many Cox proportional hazards models from which the c.d.f. can be recovered. Specifying `loc` fits a location model, and specifying `locsca` estimates a location-scale model. Inference is based on a bootstrap.

We obtain conditional QTES of the effect of `suppins` on `ltotexp` at quantiles 0.25, 0.5, and 0.75, controlling for `totchr`, `age`, `white`, and `female`. We use `method(locsca)` because it is much faster computationally than, in particular, `method(qr)`. We obtain

```

. * Unconditional QTE of suppins on ltotexp using community-contributed
> command cdeco
. set seed 10101
. cdeco ltotexp totchr age white female, group(suppins) method(locsca)
>     nreg(100) reps(400) quantile(0.25(0.25)0.75)
(bootstrapping .....).
> .....
> .....
> .....
> .....
> .....
Conditional model                                location scale model
Number of regressions estimated                100
The variance has been estimated by bootstrapping the results 400 times.
No. of obs. in the reference group            1207
No. of obs. in the counterfactual group      1748

```

Differences between the observable distributions (based on the conditional model)

Quantile	Quantile effect	Pointwise Std. Err.	Pointwise [95% Conf. Interval]		Functional	
			[95% Conf. Interval]	[95% Conf. Interval]	[95% Conf. Interval]	[95% Conf. Interval]
.25	-.362816	.063507	-.487287	-.238344	-.499034	-.226598
.5	-.247377	.054152	-.353514	-.14124	-.36353	-.131224
.75	-.137208	.055354	-.245701	-.028716	-.255939	-.018478

Effects of characteristics

Quantile	Quantile effect	Pointwise Std. Err.	Pointwise [95% Conf. Interval]		Functional	
			[95% Conf. Interval]	[95% Conf. Interval]	[95% Conf. Interval]	[95% Conf. Interval]
.25	.002096	.028978	-.0547	.058892	-.066256	.070447
.5	-.013416	.027461	-.067239	.040406	-.078189	.051357
.75	-.023122	.028571	-.07912	.032876	-.090513	.044269

Effects of coefficients

Quantile	Quantile effect	Pointwise Std. Err.	Pointwise [95% Conf. Interval]		Functional	
			[95% Conf. Interval]	[95% Conf. Interval]	[95% Conf. Interval]	[95% Conf. Interval]
.25	-.364911	.061113	-.484691	-.245132	-.504837	-.224986
.5	-.233961	.051964	-.335809	-.132113	-.35294	-.114982
.75	-.114086	.056582	-.224984	-.003188	-.243637	.015464

Bootstrap inference on the counterfactual quantile processes

Null-hypothesis	P-values	
	KS-statistic	CMS-statistic
Correct specification of the parametric model 0	.0025	.0075
Correct specification of the parametric model 1	0	0
Differences between the observable distributions		
No effect: $QE(\tau)=0$ for all τ s	0	0
Constant effect: $QE(\tau)=QE(0.5)$ for all τ s	.0025	0
Stochastic dominance: $QE(\tau)>0$ for all τ s	0	0
Stochastic dominance: $QE(\tau)<0$ for all τ s	.6825	.6825
Effects of characteristics		
No effect: $QTE(\tau)=0$ for all τ s	.5975	.6675
Constant effect: $QE(\tau)=QE(0.5)$ for all τ s	.2725	.27
Stochastic dominance: $QE(\tau)>0$ for all τ s	.31	.33
Stochastic dominance: $QE(\tau)<0$ for all τ s	.56	.56
Effects of coefficients		
No effect: $QE(\tau)=0$ for all τ s	0	0
Constant effect: $QE(\tau)=QE(0.5)$ for all τ s	0	0
Stochastic dominance: $QE(\tau)>0$ for all τ s	0	0
Stochastic dominance: $QE(\tau)<0$ for all τ s	.7075	.7075

The first set of output gives the conditional QTES that have reverse sign to those found in the preceding examples, because the `cdeco` command sets the reference group to `suppins=1`, but are of similar magnitude. For example, the method of [Firpo \(2007\)](#) yielded QTES of, respectively, 0.399, 0.283, and 0.149, and the RIF method yielded values of, respectively, 0.399, 0.273, and 0.162. The output reports both pointwise confidence intervals and uniform confidence intervals. The `cdeco` results do vary numerically, though not qualitatively, with the `method()` option used; `method(logit)` has the attraction of being relatively fast to compute compared with `method(qr)`.

The next set of tables decomposes the QTES into the effects of characteristics and the effects of coefficients. The effect of variation in characteristics across individuals is not great.

The final table presents a variety of specification tests. The first tests reject the underlying location-scale model at significance level 0.05, so other methods for estimating the conditional c.d.f. might be used.

25.9.4 Unconditional QTE with endogenous discrete binary regressor

[Frölich and Melly \(2010, 2013\)](#) provide an estimator for the case in which the binary treatment variable D is endogenous and a binary instrumental variable z is available. Several assumptions are made, including that the regression controls for all possible confounders.

The objective function is the same as (25.11) for unconditional QTE with an exogenous treatment, except the weights are now given by

$$w_i = \frac{z_i - \hat{p}(z_i)}{\hat{p}(z_i)\{1 - \hat{p}(z_i)\}}(2D_i - 1)$$

where $\hat{p}(z_i)$ is a prediction of $\Pr(z_i = 1 | \mathbf{x}_i)$.

This model can be fit by the `ivqte` command; the command syntax is as for conditional QTE with a binary instrument, but the option `aai` is dropped.

```
. * Unconditional QTE of endogenous suppins using ivqte
. ivqte ltotexp (suppins=marry), dummy(white female) continuous(age totchr)
> variance trim(0.01) quantile(0.5)

Unconditional Quantile Treatment Effects under endogeneity
Estimator suggested in Froelich and Melly (2008)

Quantile(s): .5
Dependent variable: ltotexp
Treatment variable: suppins
Instrumental variable: marry
Control variable(s): age totchr white female
Number of observations: 2955
Proportion of compliers: .077

Propensity score estimated by local logit regression with h = infinity and lambda
> = 1
Variance estimated using local logit regression with h = infinity and lambda = 1
```

ltotexp	Coefficient	Std. err.	z	P> z	[95% conf. interval]
Quantile_1	.5061746	.8248461	0.61	0.539	-1.110494 2.122843

At the median, the TE due to supplementary insurance is nearly 51%, but the confidence interval is very wide and includes 0. A better instrument is needed, as is also clear from the proportion of compliers being only 0.077.

Repeating this estimation for the other quartiles, we obtain estimated unconditional local QTES of, respectively, 0.087, 0.506, and 1.292 at the 0.25, 0.5, and 0.75 quantiles. By comparison, the corresponding estimated conditional local QTES with endogeneity were 1.374, 1.179, and 1.032.

25.10 Additional resources

The treatment evaluation methods of this chapter are presented in detail in [Angrist and Pischke \(2009\)](#), [Cunningham \(2021\)](#), and [Glewwe and Todd \(2022\)](#); see also [Cameron and Trivedi \(2005](#), chap. 25), [Wooldridge \(2010](#), chap. 21), and [Hansen \(2022](#), chaps. 18, 21, and 24). The website for [Cunningham \(2021\)](#) includes associated Stata programs and examples.

The Stata ERM commands such as `eregress` and `eprobit` estimate TEs in linear and leading nonlinear models with treatment that is exogenous or endogenous. The Stata `et` commands such as `etregress` and `eteffects` cover ET binary treatment modeled by a probit model. For ET, these commands require an instrument, require specification of functional forms, including the way in which any TE heterogeneity occurs, and can require assumptions as strong as joint normality of model errors. For linear models with heterogeneous responses, the community-contributed `ivtreatreg` command of [Cerulli \(2014\)](#) deals with specific types of heterogeneity.

[Abadie and Cattaneo \(2019\)](#) and [Huber \(2019\)](#) provide excellent comprehensive surveys that emphasize quasi-experimental methods with endogenous treatment.

This chapter provides only simple examples of the various quasi-experimental methods with endogenous treatment. [Imbens and Rubin \(2015\)](#) cover LATE, and [Huber and Wüthrich \(2019\)](#)

The various causal methods of this chapter rely on strong assumptions. These can be partly testable using falsification tests, placebo tests, and graphical tools that vary with the causal method used. For brevity, these have not been emphasized here but should be included in any application.

The topic of quasi-experimental methods is a very active area of research. The methods presented in this chapter are currently being refined and extended.

25.11 Exercises

1. For the multilevel treatment example of section [25.3.3](#), perform separate analysis for men and women (note that variable `h_male` needs to be dropped as a regressor). Does the ATE differ by gender?
Comment.
2. In section [25.4.2](#), the `etregress` command is used to estimate the TE of insurance using three different estimators. In each case, the model being fit uses `ssiratio` as the single instrument and hence is just identified. Now, fit an overidentified model with an additional instrument by omitting the variable `linc` from the outcome equation for `ldrugexp`, but designate it as a second excluded instrument. Refit the overidentified model using all three methods, and compare the estimated TEs with those from the just-identified model.
3. In exercise 2 above, the sample pools male and female subjects. The model specification allows for heterogeneity in response up to an intercept shift. The estimated TE is the same for both sexes. Consider the hypothesis that the TEs differ between male and female subjects. Using the specification of section [25.4.2](#), estimate the TEs for male and female groups, and compare them.
4. The TE estimate from the linear specification used in section [25.4.2](#) is directly available from the regression estimate. Consider the following nonlinear-in-variables specification that includes an additional interaction variable generated by the product of `hi_empunion` and `female`. What additional complications of estimation and interpretation arise in this case? Which variant of the `etregress` command would be suitable for estimating the TE? Generate the interaction effect, add it to the specification, apply your estimation method, and obtain estimates of the TE using the `margins` command.
5. Emergency room visits are expensive, and many could be avoided through better access to regular doctors. Apply the methods of section [25.5.4](#) to outcome variable `ervisits`. Compare the OLS estimate of the TE with the LATE estimate. Has increased access to Medicaid had the desired effect?
6. Repeat the synthetic control example of section [25.6.2](#) with outcome `lncigsale`, the natural logarithm of `cigsale`. Note that you should

also change the control variables in the `synth` command to `lncigsale(1988)`, `lncigsale(1980)`, and `lncigsale(1975)`. Compared with the analysis with outcome `cigsale`, is there a change in the states with nonzero weight? Are the postoutcome TES of comparable magnitude when analysis is in logs rather than levels of cigarette sales?

7. Generate data on `y`, `x`, and `D` using the code given at the start of section [25.7.2](#). Apply the `rdplot` and `rdrobust` commands, using command defaults, and compare your results with those obtained in section [25.7.2](#).
8. Generate data on `y`, `x`, and `D` using the code given at the start of section [25.7.2](#), with the change that `y` is generated with error that is `rnormal(0, 10)` rather than `rnormal(0, 10)`. Obtain the SRD estimate of the TE using the `rdrobust` command, and note that estimate precision has been greatly increased. Next, create an FRD design by redefining `D` as `D=x+runiform(-10, 10)>0`. Use commands `rdplot`, `rdrobust`, and then `rdrobust` with the option `fuzzy(D)`. Comment on your results.

Chapter 26

Spatial regression

26.1 Introduction

Spatial regression models are models for observations that are related, with the strength of this relation decreasing as the distance between observations increases.

The term “distance” is used quite broadly. The Stata documentation focuses on geospatial data for which the distance measure may be physical distance or on whether observational units share a common boundary. But the Stata spatial regression commands are applicable for any distance measure, such as economic distance or social distance or peer group membership or network ties such as friendships.

In the simplest spatial regression models, only error terms are spatially correlated. This complication is comparable with one of autocorrelated errors or clustered errors. Then the usual estimators can be used, if error terms are uncorrelated with regressors, but valid statistical inference requires that the standard errors of parameter estimates be corrected to control for the spatial correlation in the errors. More efficient estimation, such as feasible generalized least squares (FGLS), is possible if we additionally specify a particular model for the error correlation.

A much greater complication arises if the dependent variable for one observation depends directly on the values of the dependent variable for nearby observations. This is comparable with autoregressive time-series models where the dependent variable in the current period depends directly on values of the dependent variable in previous periods. The spatial relationship is represented by one or more spatial weighting matrices \mathbf{W} .

Spatial regression presents unique complications. The key is forming any spatial weighting matrices and specifying the pathway of spatial correlation that could be via the dependent variable of nearby individuals, regressors of nearby individuals, or errors of nearby individuals. The Stata spatial commands and documentation focus on forming weighting matrices \mathbf{W} from geospatial data that provide longitudinal and latitudinal

coordinates. But the user can also provide his or her own \mathbf{W} matrix where necessary, such as with peer-effects data and network data.

26.2 Overview of spatial regression models

The spatial autoregressive (SAR) in the (conditional) mean model specifies

$$y_i = \mathbf{x}'_i \boldsymbol{\beta} + \lambda \times \sum_{j=1}^N w_{ij} y_j + u_i$$

where $w_{ii} = 0$, so y_i does not appear in the right-hand side. In matrix notation, we have

$$\mathbf{y} = \lambda \mathbf{W} \mathbf{y} + \mathbf{X} \boldsymbol{\beta} + \mathbf{u}$$

where \mathbf{W} is an $N \times N$ spatial weighting matrix that has diagonal entries zero. The term “SAR” arises by analogy to the time-series autoregressive model $y_t = \rho y_{t-1} + u_t + \mathbf{x}'_t \boldsymbol{\beta}$, and the additional term “ $\mathbf{W} \mathbf{y}$ ” is called a spatial lag (in the dependent variable).

The crucial ingredient is specification of the weights w_{ij} . In principle, these could be a given function of distance that depends in part on unknown parameters that are estimated. Instead, the `sp` commands use as weights constants that need to be specified by the researcher. For different types of spatial data, there are different commonly used methods for specifying the spatial weights. For example, in the standard peer-effects model, it is assumed that only peers matter and that the peer effect is the average of one’s peers. If person 1 has persons 2, 3, and 4 as his or her only peers, then $y_1 = \mathbf{x}'_1 \boldsymbol{\beta} + \lambda \{(y_2 + y_3 + y_4)/3\} + u_1$, so $w_{12} = w_{13} = w_{14} = 1/3$ and all other $w_{1j} = 0$.

A different form of spatial dependence is solely through the error term. A SAR in the error model of order one specifies $u_i = \sum_{j=1, j \neq i}^N w_{ij} u_j + \varepsilon_i$, so

$$\begin{aligned}\mathbf{y} &= \mathbf{X}\boldsymbol{\beta} + \mathbf{u} \\ \mathbf{u} &= \rho\mathbf{W}_u\mathbf{u} + \boldsymbol{\varepsilon}\end{aligned}$$

where the underlying errors $\boldsymbol{\varepsilon}_i$ are assumed to be independent.

The Stata SAR models allow spatial dependence through the conditional mean, model errors, and regressors. The SARAR(1,1) model, with one spatial lag in \mathbf{y} and one spatial lag in \mathbf{u} , as well as some spatial dependence through the regressors, specifies

$$\begin{aligned}\mathbf{y} &= \lambda\mathbf{W}_y\mathbf{y} + \mathbf{X}\boldsymbol{\beta} + \gamma\mathbf{W}_x\mathbf{x}_p + \mathbf{u} \\ \mathbf{u} &= \rho\mathbf{W}_u\mathbf{u} + \boldsymbol{\varepsilon}\end{aligned}$$

where the single regressor \mathbf{x}_p is usually a regressor contained in \mathbf{X} . The errors $\boldsymbol{\varepsilon}_i$ are assumed to be independent and, in the case of maximum likelihood (ML) estimation, independent and identically distributed (i.i.d.) normal. Not all the three spatial lags need to be included in this model, and where more than one of these spatial lags appears, a common spatial weighting matrix \mathbf{W} may be used.

Even more general SARAR(p, q) models may add additional spatial lags. For example, we may have two spatial lags in the mean ($p = 2$), so

$$\mathbf{y} = \lambda_1\mathbf{W}_1\mathbf{y} + \lambda_2\mathbf{W}_2\mathbf{y} + \mathbf{X}\boldsymbol{\beta} + \gamma\mathbf{W}_x\mathbf{X}_p + \mathbf{u}$$

The Stata SAR commands focus on these SARAR models. The `spregress` command therefore covers models that allow for spatial dependence through the dependent variable, through the errors, and through the regressors. If the explanatory variables are endogenous and instruments are available, then the `spivregress` command can be used. And for data with multiple observations per unit, such as panel data, one can use the `spxtregress` command.

26.3 Geospatial data

The [SP] *Stata Spatial Autoregressive Models Reference Manual* uses data on the homicide rate in 1990 in counties in the southern United States. We use the same data, except analysis is restricted to the 159 counties in the state of Georgia.

26.3.1 Spatial dataset

The key dataset on homicides in Georgia is

Variable Storage Display Value					
name	type	format	label	Variable label	
_ID	int	%12.0g	Spatial-unit ID		
_CX	double	%10.0g	x coordinate of area centroid		
_CY	double	%10.0g	y coordinate of area centroid		
cname	str20	%20s	County name		
hrate	double	%12.10f	Homicide rate per 100,000		
ln_population	double	%12.10f	Log of population size		
poverty	double	%12.10f	Percentage of families below poverty line		
. summarize _ID _CX _CY cname hrate ln_population poverty					
Variable	Obs	Mean	Std. dev.	Min	Max
_ID	159	2424.365	179.4522	2100	2722
_CX	159	-83.57566	1.039462	-85.50329	-81.15262
_CY	159	32.8082	1.18595	30.70841	34.91453
cname	0				
hrate	159	12.39901	6.723074	0	37.55566
ln_population	159	9.872061	1.076768	7.557473	13.38311
poverty	159	15.49396	6.097446	1.916347	30.41627

There are 159 observations, one for each county. The dependent variable of interest is `hrate`, the homicide rate per 100,000 people in 1990. Explanatory variables will be the log of population and the poverty rate. The variable `_ID` is the unique code for each county. The variables `_CX` and `_CY` give the two-dimensional geographic coordinates of the centroid of each county.

The variables `_ID`, `_CX` and `_CY` were created by earlier use of the `spset` command, which is detailed in [SP] **spset**. We have

```
. * Attach to correct shapefile using spset, modify, and save
. spset, modify shpfile(mus226georgia_shp)
  (creating _ID spatial-unit id)
  (creating _CX coordinate)
  (creating _CY coordinate)

  Sp dataset: mus226georgia.dta
Linked shapefile: mus226georgia_shp.dta
  Data: Cross sectional
  Spatial-unit ID: _ID
  Coordinates: _CX, _CY (planar)

. save mus226georgia, replace
file mus226georgia.dta saved
```

The coordinates are planar, rather than degrees of latitude and longitude. This distinction is explained below in the subsection on measuring distance between observations. The dataset is linked to a shapefile, which we consider next.

26.3.2 Geospatial shapefile

Geospatial analysis can be done using just the preceding dataset, which includes the centroids of each county that can be used to obtain distances between each county.

For some geospatial analysis, however, it is beneficial to additionally have shapefiles that detail the boundaries of each county. In particular, this enables construction of heat maps that indicate the values of a variable in each region, and it enables the spatial weighting matrix \mathbf{W} to be a contiguity matrix for which w_{ij} is positive if observations i and j are in adjoining regions.

Shapefiles in standard format can be obtained from sources on the web. These then need to be converted to a Stata format shapefile, using the `spshape2dta` command that links the shapefile and the original data file; for details, see [SP] **Intro 4**.

The associated Stata format shapefile for the current example is

```
. * Read in shapefile with coordinates for Georgia counties and summarize
. qui use mus226georgia_shp, clear
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
_ID	4,737	2430.068	179.2355	2100	2722
_X	4,576	-83.52931	1.063287	-85.60899	-80.89492
_Y	4,576	32.75887	1.188491	30.36106	35.00028
rec_header	0				
shape_order	4,737	17.00591	11.12619	1	58

Here 4,576 sets of coordinates are used to define the boundaries of the 159 counties.

For example, for the county with county identifier 2108, we have

```
. * List the county boundary coordinates for county 2108
. list _ID _X _Y shape_order if _ID == 2108, clean
```

	_ID	_X	_Y	shape_~r
190.	2108	.	.	1
191.	2108	-83.776512	34.788391	2
192.	2108	-83.813553	34.891724	3
193.	2108	-83.849831	34.888264	4
194.	2108	-83.864494	34.900963	5
195.	2108	-83.907837	34.914978	6
196.	2108	-83.921082	34.943623	7
197.	2108	-83.939964	34.961624	8
198.	2108	-83.937996	34.989391	9
199.	2108	-83.549416	34.989536	10
200.	2108	-83.551826	34.944477	11
201.	2108	-83.590721	34.936024	12
202.	2108	-83.603844	34.90472	13
203.	2108	-83.663338	34.870159	14
204.	2108	-83.654549	34.814987	15
205.	2108	-83.679329	34.794216	16
206.	2108	-83.709503	34.780991	17
207.	2108	-83.733513	34.791134	18
208.	2108	-83.776512	34.788391	19

The boundaries of county 2108 are defined using 18 (X, Y) coordinates.

26.3.3 Heat maps

The `grmap` command produces a heat map or choropleth map that presents a geographic map with different shadings used to display different values of the variable of interest.

For the homicide rate, we obtain

```
. * Provide a heat map for homicide rates in Georgia counties
. qui use mus226georgia, clear
. grmap, activate
grmap already activated
. grmap hrate, clmethod(custom) clbreaks(0 10 20 40) legend(pos(1)) fcolor(Greys)
```

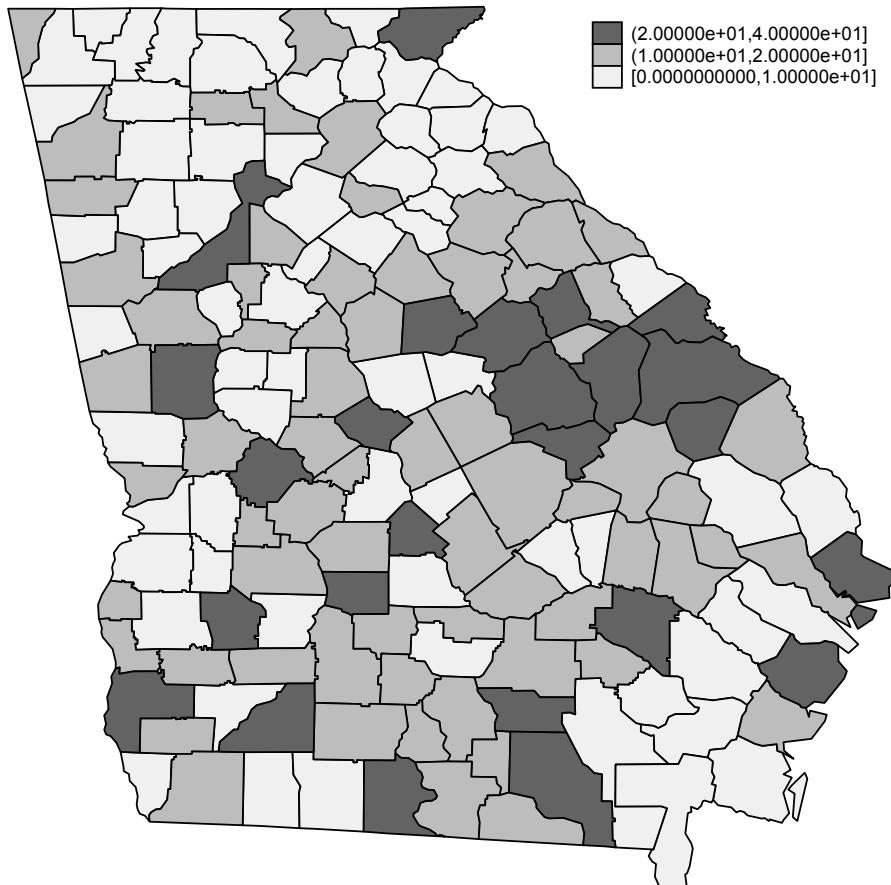


Figure 26.1. Heat map of homicide rate in various counties

The resulting map is given in figure [26.1](#). The darkest shaded area has the highest homicide rates, in excess of 20 homicides per 100,000 people. For example, the darker area at three o'clock on the map covers counties in the Augusta region. There does appear to be some spatial correlation because

there are several clusters of dark-shaded counties and several clusters of lightly shaded counties.

26.3.4 Geospatial distance

Geospatial data are often located using latitude, measured in degrees north or south of the equator, and longitude, measured in degrees west or east of the north-south meridian passing through the British Royal Observatory in Greenwich.

In computing Euclidean distance between individuals, one cannot immediately use degrees of latitude and longitude. For example, a movement directly east of one degree of longitude at the equator is a movement of approximately 69 miles, while at 80 degrees north, it is a movement of only 12 miles.

If `sp` datasets use latitude and longitude, then `_cx` is longitude and `_cy` is latitude (note the reverse ordering). The current dataset appears to use latitude and longitude because, for example, Atlanta, Georgia, has coordinates (33.7490°N, 84.3880°W). However, this is not the case, because the earlier `spset` command stated that planar coordinates are used instead. Apparently, the planar coordinates have been scaled to be approximately the same as the latitude and longitude coordinates for the Georgia region.

The `coordsys()` option of the `spset` command can set the coordinate system to be planar or latitude and longitude. The `spdistance` command computes the Euclidean distance between two observations, with appropriate adjustment for the earth's curvature if latitude and longitude is used instead of planar coordinates.

For the current example with planar coordinates, we compute the distance between counties 2100 and 2103. We have

```
. * Compute Euclidean distance between observations using planar coordinates
. spdistance 2100 2103
(data currently use planar coordinates)
      _ID      (x, y) (planar)

```

2100	(-83.40281, 34.87811)
2103	(-84.96217, 34.80377)

```
      distance      1.5611309 planar units
. display "Distance = " sqrt((-83.40281-(-84.96217))^2+(34.87811-34.80377)^2)
Distance = 1.561131
```

26.4 The spatial weighting matrix

Compared with basic regression, the only extra information needed to use Stata's `sp` regression commands is the spatial weighting matrix \mathbf{W} . The matrix \mathbf{W} can be created elsewhere and stored on an external file or created using community-contributed `mata` code or created using `sp` commands. In all cases, the weighting matrix is ultimately input to Stata using one of the many variants of the `spmatrix` command.

26.4.1 Creating a spatial weighting matrix

Here we create \mathbf{W} in the special case of geospatial data by directly using the `spmatrix create` command. This requires a Stata dataset that has been `spset`. The `spmatrix create` command creates a spatial weighting matrix that may be either an inverse-distance matrix or a contiguity matrix that is based on sharing borders.

For an inverse-distance matrix, one uses the `spmatrix create idistance` command. Then the weight w_{ij} is inversely proportional to the distance between i and j . The `vtruncate(#)` option truncates to zero for distances greater than $#$. This command requires coordinates for each observation but does not require a shapefile.

For a contiguity matrix, the default for the `spmatrix create contiguity` command is to set w_{ij} to 1 (before normalization) if i and j share a border or a vertex and to set w_{ij} to 0 otherwise. Various options instead set w_{ij} to 0 if the only thing shared is a vertex (rather than a border) or, more expansively, also set w_{ij} to 1 (before normalization) if i and j are neighbors of neighbors. This command requires a shapefile in addition to coordinates for each observation.

In either case, the matrix \mathbf{W} is normalized, using the `normalize()` option. The default option is `normalize(spectral)`, which rescales every entry in the unnormalized spatial matrix by multiplication by a constant so that the largest eigenvalue of \mathbf{W} equals 1. The `normalize(minmax)` option rescales every entry in the unnormalized spatial matrix by division by a

constant equal to the smallest of the largest row sum or the largest column sum. The `normalize(row)` option multiplies all entries in each row by a constant to ensure that the entries in each row sum to 1, so $\sum_{j=1}^N w_{ij} = 1$. The `normalize(none)` option leaves the matrix unnormalized.

Because $\lambda \mathbf{W} = (\lambda/a) \times (a\mathbf{W})$, a normalization that multiplies all entries in \mathbf{W} by a common constant is innocuous; it changes the associated scalar parameter only from λ to λ/a . The remaining estimates after commands such as `spregress` are unchanged. In the case of row normalization, however, all estimates will change because different rescalings have been applied to different rows of \mathbf{W} .

For the current example, a contiguity weighting matrix is obtained, using the default spectral normalization. We have

```
. * Create & summarize weighting matrix W - contiguity with spectral normalization
. spmatrix create contiguity W
. spmatrix summarize W
```

Weighting matrix W

Type	contiguity
Normalization	spectral
Dimension	159 x 159
Elements	
minimum	0
minimum > 0	.1651123
mean	.0055906
max	.1651123
Neighbors	
minimum	1
mean	5.383648
maximum	10

The 159×159 matrix \mathbf{W} has entries that are either 0 or 0.1651123. Counties have on average 5.38 neighbors, where a neighboring county is one that shares a border or vertex. Counties have between 1 and 10 neighbors, so the row sums of \mathbf{W} range from 0.1651123 to 1.651123. Further details on the matrix \mathbf{W} can be obtained by using the `spmatrix matafromsp` command to pass \mathbf{W} to `mata` and then using relevant `mata` matrix commands.

At this stage, the user should think seriously about whether this is a reasonable weighting matrix. Suppose county i has three neighbors, counties

m , n , and o . Then, ignoring regressors and errors, the SAR in mean model specifies $y_i = \lambda(0.165 \times y_m + 0.165 \times y_n + 0.165 \times y_0)$. By contrast, if county j had only the 1 neighbor, county m , we have $y_j = \lambda(0.165 \times y_m)$. The weights impose the restriction that homicides in neighboring counties have less impact when there are fewer neighboring counties. A better model may specify that we use the average of the homicides over adjoining counties. Then $y_i = \delta \times (y_m + y_n + y_0)/3$ and $y_j = \delta \times y_m$. This corresponds to applying a row normalization, the `normalize(rows)` option, rather than the default spectral normalization to the contiguity matrix.

The importance of the specification of the spatial weights cannot be overemphasized. The weights determine any spatial spillovers. And consistent estimation requires correct model specification, including specification of the spatial weighting matrix.

26.4.2 Creating spatial lag variables

Given a spatial weighting matrix \mathbf{W} , spatial lags of variables such as $\mathbf{W}\mathbf{y}$ and $\mathbf{W}\mathbf{x}$ can be constructed using the `spgenerate` command.

For example, the spatial lag of the homicide rate is

```
. * Create spatial lag Wy
. spgenerate Whrate = W*hrate
. summarize hrate Whrate
```

Variable	Obs	Mean	Std. dev.	Min	Max
hrate	159	12.39901	6.723074	0	37.55566
Whrate	159	11.25052	4.591557	.7547128	27.27849

	hrate	Whrate
hrate	1.0000	
Whrate	0.2229	1.0000

The spatially lagged variable is weakly correlated with the unlagged variable.

26.5 OLS regression and test for spatial correlation

Ordinary least-squares (OLS) regression of the homicide rate on log population and the poverty rate yields

```
. * OLS estimation using regress
. regress hrate ln_population poverty, vce(robust)

Linear regression
Number of obs      =      159
F(2, 156)          =     21.33
Prob > F          =     0.0000
R-squared          =     0.2326
Root MSE           =     5.9273
```

hrate	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
ln_population	1.614672	.6396782	2.52	0.013	.3511237	2.878221
poverty	.6349861	.1013695	6.26	0.000	.4347522	.83522
_cons	-13.37958	7.453859	-1.79	0.075	-28.10309	1.343937

A 1% increase in the population is associated with 0.016 more homicides per 100,000 people, and an increase in the poverty rate by 1 percentage point is associated with a substantial 0.63 increase in the homicide rate (the mean homicide rate is 12.39). These effects are statistically significant at level 0.05.

A standard test for spatial correlation is the Moran I test. The distribution of the test statistic is obtained under the null hypothesis that $y_i = \mathbf{x}'_i \beta + u_i$, where u_i are i.i.d. $(0, \sigma^2)$. The test requires that a spatial weighting matrix \mathbf{W} be specified under the alternative hypothesis. The test is asymptotically equivalent to a score test of an SAR(1) in mean model with spatial weighting matrix \mathbf{W} or of an SAR(1) in error model with spatial weighting matrix \mathbf{W} , against a model with no spatial dependence.

The test is implemented using the `estat moran` command. We have

```
. * Moran test for spatial correlation following OLS
. qui regress hrate ln_population poverty
. estat moran, errorlag(W)
Moran test for spatial dependence
      H0: Error terms are i.i.d.
      Errorlags: W
      chi2(1)      =      0.59
      Prob > chi2  =  0.4417
```

The test statistic has $p = 0.44 > 0.05$, so we do not reject the null hypothesis of no spatial correlation at level 0.05. There is little evidence of spatial correlation in these data, at least when we use the current spatial weighting matrix \mathbf{W} . So we expect the subsequent analysis using spatial regression commands with this specification of \mathbf{W} to yield results similar to OLS.

26.6 Spatial dependence in the error

When there is spatial dependence only in the error, a number of methods have been proposed to obtain correct standard errors and to obtain more efficient estimates.

26.6.1 Spatial heteroskedastic- and autocorrelation-consistent standard errors for OLS

OLS of y_i on \mathbf{x}_i remains consistent when the only spatial dependence is in the error. Spatial heteroskedastic- and autocorrelation-consistent (HAC) standard errors provide spatial robust standard errors analogous to, for example, HAC standard errors for time-series data. These standard errors are nonparametric in that they do not require an explicit model for the correlation in model errors.

In general, the variance matrix of the OLS estimator is

$$\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}}) = (\mathbf{X}'\mathbf{X})^{-1} \left\{ \sum_{i=1}^N \sum_{j=1}^N E(\mathbf{x}_i \mathbf{x}_j' u_i u_j) \right\} (\mathbf{X}'\mathbf{X})^{-1}$$

where the only terms in the middle matrix that contribute are the terms for which $E(\mathbf{x}_i \mathbf{x}_j' u_i u_j) \neq 0$. In the spatial context, $E(\mathbf{x}_i \mathbf{x}_j' u_i u_j) \rightarrow 0$ as the distance between individuals i and j grows.

In the case of a one-dimensional distance measure d_{ij} , with $E(\mathbf{x}_i \mathbf{x}_j' u_i u_j) = 0$ for $d_{ij} > \delta$, a simple spatial HAC variance estimate is

$$\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}}) = (\mathbf{X}'\mathbf{X})^{-1} \left\{ \sum_{i=1}^N \sum_{j=1}^N \mathbf{1}(d_{ij} < \delta) (\mathbf{x}_i \mathbf{x}_j' \widehat{u}_i \widehat{u}_j) \right\} (\mathbf{X}'\mathbf{X})^{-1}$$

where it is assumed that $(1/N^2)\sum_i\sum_j \mathbf{1}(d_{ij} < \delta) \rightarrow 0$ as $N \rightarrow \infty$. More generally, kernel weights may be used in place of $\mathbf{1}(d_{ij} < \delta)$. This estimate generalizes straightforwardly to instrumental-variables (IV) and generalized method of moments estimation.

[Conley \(1999\)](#) proposed this estimator for generalized method of moments estimation. He more generally allowed for a multidimensional distance measure, such as (X, Y) coordinates for geospatial data. Then data are viewed as being arranged on a lattice. [Kelejian and Prucha \(2007\)](#) present a similar HAC estimate for OLS and IV estimation, using alternative assumptions regarding the stochastic process for spatial correlation.

We obtain HAC standard errors using the estimate of [Conley \(1999\)](#). In the one-dimensional coordinate case with coordinate C_i , the distance $d_{ij} = C_i - C_j$, and Conley uses the kernel weight $\mathbf{1}(d_{ij} < \delta) \times (1 - d_{ij}/\delta)$ that has declining weights rather than the simpler $\mathbf{1}(d_{ij} < \delta)$ that has uniform weights. In the two-dimensional case, with distances $d_{1ij} = C1_i - C1_j$ and $d_{2ij} = C2_i - C2_j$, we use the kernel weight $\mathbf{1}(d_{1ij} < \delta_1)\mathbf{1}(d_{2ij} < \delta_2) \times (1 - d_{1ij}/\delta_1)(1 - d_{2ij}/\delta_2)$.

The community-contributed `x_ols` command ([Dube 1999](#)) implements this method. It requires passing, in turn, the two variables with the X and Y coordinates, the distance cutoffs for the two coordinates, the dependent variable, the regressors (including an intercept if relevant), and options giving the number of regressors (including an intercept if relevant) and the number of coordinates.

```
. * OLS with Conley spatial HAC standard errors using user addon x_ols
. generate const = 1
. generate cutoff1 = 1.04 // One standard deviation of _CX
. generate cutoff2 = 1.19 // One standard deviation of _CY
. x_ols _CX _CY cutoff1 cutoff2 hrate ln_population poverty const, xreg(3) coord(2)
```

Results for Cross Sectional OLS corrected for Spatial Dependence

number of observations= 159

Dependent Variable= hrate

variable	ols estimates	White s.e.	s.e. corrected for spatial dependence
ln_population	1.6146721	.5273927	.57420456
poverty	.63498613	.09313404	.1010521
const	-13.379577	6.1467955	6.689321

The cutoffs were chosen to be the standard deviations of, respectively, `_CX` and `_CY`. The standard errors denoted `white s.e.` are actually default nonrobust standard errors based on $s^2(\mathbf{X}'\mathbf{X})^{-1}$. Compared with the heteroskedastic–robust standard errors given in earlier output, the spatial HAC standard errors are slightly smaller.

An alternative HAC estimate for this geospatial data example would use a single distance measure d_{ij} . This could, for example, be the Euclidean distance $d_{ij} = \sqrt{(C1_i - C1_j)^2 + (C2_i - C2_j)^2}$. The `x_ols` command does not directly cover this case because it takes as inputs N coordinates such as $(C1_i, C2_i)$, $i = 1, \dots, N$, rather than N^2 distances d_{ij} , $i = 1, \dots, N$, $j = 1, \dots, N$.

26.6.2 FGLS estimation

More efficient FGLS estimation is possible if a parametric model for $\text{Cov}(u_i, u_j | \mathbf{X})$ is specified.

The SAR in error model fit by the `spregress` command (see section [26.7](#)) is one such model. Here we briefly mention some other models.

The negative exponential distance decay model specifies $E(\mathbf{u}\mathbf{u}') = \sigma^2(\mathbf{I} + \delta\Psi)$, where $\Psi_{ij} = \exp(-\gamma d_{ij})$.

The common shocks model with m common shocks specifies $\mathbf{u} = \mathbf{Hv} + \boldsymbol{\varepsilon}$, where \mathbf{v} is $m \times 1$, \mathbf{H} is an $N \times m$ weighting matrix, and the underlying errors v_i and ε_i are i.i.d. Then $E(\mathbf{u}\mathbf{u}') = \sigma_v^2 \mathbf{H}\mathbf{H}' + \sigma_\varepsilon^2 \mathbf{I}$, and estimation is straightforward because a closed-form solution exists for $(\sigma_v^2 \mathbf{H}\mathbf{H}' + \sigma_\varepsilon^2 \mathbf{I})^{-1}$.

A factor model assumes observations aggregate to a super region g (group) with common shocks for observations in region g . Then $y_{ig} = \mathbf{x}'_{ig}\boldsymbol{\beta} + u_{ig}$, $u_{ig} = \delta_i f_g + \varepsilon_{ig}$, δ_i is a factor loading, and f_g is a factor. This can be viewed as a special case of the common shocks model.

26.7 Spatial autocorrelation regression models

If spatial correlation is only in the error term, then OLS is consistent, and the only challenges are to obtain standard errors that control for spatial correlation or to obtain FGLS estimates that are more efficient than OLS.

If instead spatial correlation appears as a spatial lag in the dependent variable, then OLS is inconsistent, so alternative estimation methods are needed.

For example, suppose the first two observations are dependent on each other, with 1) $y_1 = \lambda y_2 + u_1$ and 2) $y_2 = \lambda y_1 + u_2$. Then from 2) y_2 depends on y_1 , which from 1) depends on u_1 . It follows that y_2 is correlated with u_1 , so in 1) the explanatory variable y_2 is correlated with the error u_1 , leading to OLS being inconsistent. This problem arises even if the errors u_1 and u_2 are independent.

Consistent estimates can be obtained by IV estimation, using values of regressors for neighbors as instruments, or by ML estimation under the stronger assumption of i.i.d. errors. The `spregress` command implements these approaches for various SAR models.

26.7.1 The `spregress` command

The `spregress` command has syntax

```
spregress depvar [indepvars] [if] [in], gs2s1s [gs2sls_options]  
spregress depvar [indepvars] [if] [in], ml [ml_options]
```

The most important options determine which of two possible estimation procedures are used. The default `gs2s1s` option provides generalized spatial two-stage least-squares (GS2SLS) estimates under the assumption that errors are independent, albeit potentially heteroskedastic if the additional `heteroskedastic` option is used. The alternative `ml` option provides ML estimates under the assumption that errors are i.i.d. normal. Note that, unlike the usual OLS and two-stage least-squares (2SLS) commands, the spatial

estimators become inconsistent if the preceding assumptions on the errors do not hold and the assumptions for the `ml` option are especially strong.

Other key options specify the nature of the spatial correlation. The `dvarlag(W)` option includes Wy , a spatial lag in the dependent variable, as a regressor. The `errorlag(W)` option includes a spatial lag Wu in the error, so $u = Wu + \varepsilon$. The `ivarlag(W: varlist)` option includes Wx_p , spatial lags in some of or all the explanatory variables as regressors.

The `gs2sls` option permits more than one spatial matrix to be specified for each of `dvarlag()`, `errorlag()`, and `ivarlag(W)`. For example, the SAR(2) in mean model includes W_1y and W_2y as regressors. The `ml` option permits more than one spatial matrix to be specified for `ivarlag()` only.

The `spregress` command without any of the options `dvarlag()`, `errorlag()`, and `ivarlag(W)` yields OLS. We use option `heteroskedastic` to obtain heteroskedastic–robust standard errors.

```
. * OLS estimation using spregress
. spregress hrate ln_population poverty, gs2sls heteroskedastic
  (159 observations)
  (159 observations (places) used)

Spatial autoregressive model
GS2SLS estimates
Number of obs = 159
Wald chi2(2) = 43.48
Prob > chi2 = 0.0000
Pseudo R2 = 0.2326
```

hrate	Coefficient	Std. err.	z	P> z	[95% conf. interval]
ln_population	1.614672	.6336148	2.55	0.011	.37281 2.856534
poverty	.6349861	.1004086	6.32	0.000	.4381888 .8317834
_cons	-13.37958	7.383205	-1.81	0.070	-27.85039 1.091237

The coefficient estimates are the same as those obtained using the `regress` command. The standard errors are $\sqrt{(N - K)/N}$ times those obtained using the `regress, vce(robust)` command, due to slightly different degrees-of-freedom adjustment.

26.7.2 SAR(1) in mean model

The SAR(1) in mean model specifies

$$\mathbf{y} = \lambda \mathbf{W}\mathbf{y} + \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$$

where it is assumed that $E(\mathbf{u}|\mathbf{X}) = \mathbf{0}$ and u_i are independent over i .

This model can be rewritten as $(\mathbf{I} - \lambda \mathbf{W})\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$, and solving for \mathbf{y} yields the reduced form

$$\mathbf{y} = (\mathbf{I} - \lambda \mathbf{W})^{-1} \mathbf{X}\boldsymbol{\beta} + (\mathbf{I} - \lambda \mathbf{W})^{-1} \mathbf{u}$$

Premultiplying the reduced-form expression for \mathbf{y} by \mathbf{W} yields

$$\mathbf{W}\mathbf{y} = \mathbf{W}(\mathbf{I} - \lambda \mathbf{W})^{-1} \mathbf{X}\boldsymbol{\beta} + \mathbf{W}(\mathbf{I} - \lambda \mathbf{W})^{-1} \mathbf{u}$$

Clearly, $\mathbf{W}\mathbf{y}$ is correlated with \mathbf{u} , and hence OLS of \mathbf{y} on $\mathbf{W}\mathbf{y}$ and \mathbf{X} in the original model will lead to inconsistent estimation even if the errors u_i are i.i.d.

GS2SLS estimation

Consistent estimates can be obtained by IV using instruments that are readily obtained; the instruments need not be variables external to the model.

Good instruments are ones that are highly correlated with $\mathbf{W}\mathbf{y}$ but are uncorrelated with \mathbf{u} . If errors are i.i.d., the optimal instrument is $E(\mathbf{W}\mathbf{y}|\mathbf{X})$. Because $E(\mathbf{u}|\mathbf{X}) = \mathbf{0}$, the preceding results imply

$$\begin{aligned} E(\mathbf{W}\mathbf{y}|\mathbf{X}) &= \mathbf{W}(\mathbf{I} - \lambda \mathbf{W})^{-1} \mathbf{X}\boldsymbol{\beta} \\ &= \mathbf{W}\mathbf{X}\boldsymbol{\beta} + \lambda \mathbf{W}^2 \mathbf{X}\boldsymbol{\beta} + \lambda^2 \mathbf{W}^3 \mathbf{X}\boldsymbol{\beta} + \dots \end{aligned}$$

provided $0 < \lambda < 1$. In principle, we could use $\mathbf{W}(\mathbf{I} - \lambda \mathbf{W})^{-1} \mathbf{X}\boldsymbol{\beta}$ as the instrument, but this requires inversion of an $N \times N$ matrix and replacing λ and $\boldsymbol{\beta}$ by consistent estimates.

Instead, we use as instruments for $\mathbf{W}\mathbf{y}$ a subset of $\mathbf{W}\mathbf{X}$, $\mathbf{W}^2\mathbf{X}$, The default for `spregress` is IV regression of \mathbf{y} on \mathbf{X} and $\mathbf{W}\mathbf{y}$ with instruments the linearly independent columns of $[\mathbf{X}, \mathbf{W}\mathbf{X}, \mathbf{W}^2\mathbf{X}]$. Higher powers such as $\mathbf{W}^3\mathbf{X}$ can be added using the `impower(#)` option.

Estimation is by 2SLS because there is more than one instrument, and inference is based on standard errors that are robust to heteroskedasticity. This estimator is implemented using the `gs2sls` option of the `spregress` command. The term “GS2SLS” is used to cover generalization detailed below for models that have a spatial lag in the error.

For the homicide rate example, we obtain

Spatial autoregressive model						Number of obs = 159
GS2SLS estimates						Wald chi2(3) = 44.27
						Prob > chi2 = 0.0000
						Pseudo R2 = 0.2329
hrate	Coefficient	Std. err.	z	P> z	[95% conf. interval]	
hrate						
ln_population	1.581835	.6481573	2.44	0.015	.3114704	2.8522
poverty	.6226799	.1087889	5.72	0.000	.4094577	.8359022
_cons	-13.29208	7.410359	-1.79	0.073	-27.81612	1.231955
W						
hrate	.0379842	.1123303	0.34	0.735	-.1821791	.2581475
Wald test of spatial terms:			chi2(1) = 0.11		Prob > chi2 = 0.7353	

The spatial lag coefficient estimate $\hat{\lambda} = 0.038$ is small and statistically insignificant, and the estimated coefficients of the regressors are quite close to the original OLS estimates.

To illustrate that GS2SLS in the case of an SAR(1) in mean model is simply 2SLS with appropriate choice of instruments, we consider the case where the instrument set goes out to only one lag. Then the instruments for $\mathbf{W}\mathbf{y}$ and \mathbf{x} are simply \mathbf{x} and $\mathbf{W}\mathbf{x}$. We create $\mathbf{W}\mathbf{x}$ regressor by regressor using the

`spgenerate` command (W_y was created earlier) and estimate using the `ivregress` command. Results are compared with the `spregress` command with option `impower(1)`. We obtain

```
. * SAR(1) in mean: Errors independent estimated using ivregress
. generate one = 1
. spgenerate Wone = W*one
. spgenerate Wln_population = W*ln_population
. spgenerate Wpoverty = W*poverty
. qui ivregress 2sls hrate ln_population poverty
>          (Whrate = W*one Wln_population Wpoverty), vce(robust)
. estimates store IVREG1
. qui spregress hrate ln_population poverty, gs2sls dvarlag(W)
>          heteroskedastic impower(1)
. estimates store SPREG1
. estimates table IVREG1 SPREG1, b(%9.4f) se eq(1)
```

Variable	IVREG1	SPREG1
#1		
Whrate	0.0314 0.1116	
ln_population	1.5875 0.6440	1.5875 0.6440
poverty	0.6248 0.1082	0.6248 0.1082
_cons	-13.3072 7.3992	-13.3072 7.3992
W		
hrate		0.0314 0.1116

Legend: b/se

The results from `ivregress` and `spregress` are identical.

ML estimation

An alternative estimator for the SAR(1) in mean model is the maximum likelihood estimator (MLE).

Under the assumption that the errors are i.i.d. $N(0, \sigma^2)$, the log likelihood is

$$\begin{aligned}\ln L(\boldsymbol{\beta}, \lambda, \sigma^2) &= \ln |\mathbf{I} - \lambda \mathbf{W}| + -\frac{N}{2} \ln 2\pi - \frac{N}{2} \ln \sigma^2 \\ &\quad - \frac{1}{2\sigma^2} (\mathbf{y} - \lambda \mathbf{W}\mathbf{y} - \mathbf{X}\boldsymbol{\beta})' (\mathbf{y} - \lambda \mathbf{W}\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\end{aligned}$$

Estimation of λ is computationally challenging for large N because $|\mathbf{I} - \lambda \mathbf{W}|$ is the determinant of an $N \times N$ matrix. Given $\hat{\lambda}_{ML}$, the ML estimate of $\boldsymbol{\beta}$ is easily obtained by OLS regression of $\mathbf{y} - \hat{\lambda}_{ML} \mathbf{W}\mathbf{y}$ on \mathbf{X} .

When errors are i.i.d. $N(0, \sigma^2)$, the usual ML results apply, provided \mathbf{W} satisfies certain restrictions. When errors are i.i.d. but not necessarily normal, then the MLE may still be consistent under some assumptions regarding the sparsity of the spatial weighting matrix; see [Lee \(2004\)](#). When errors are not i.i.d., even simply heteroskedastic, then the MLE will generally be inconsistent.

The MLE can be computed using the `ml` option. We have

```

. * SAR(1) in mean: ml with errors iid normal
. spregress hrate ln_population poverty, ml dvarlag(W)
(159 observations)
(159 observations (places) used)
(weighting matrix defines 159 places)

Performing grid search ... finished

Optimizing concentrated log likelihood:
Iteration 0: log likelihood = -506.88358
Iteration 1: log likelihood = -506.81762
Iteration 2: log likelihood = -506.81762

Optimizing unconcentrated log likelihood:
Iteration 0: log likelihood = -506.81762
Iteration 1: log likelihood = -506.81762 (backed up)

Spatial autoregressive model
Maximum likelihood estimates
Log likelihood = -506.81762
Number of obs = 159
Wald chi2(3) = 48.82
Prob > chi2 = 0.0000
Pseudo R2 = 0.2328


```

hrate	Coefficient	Std. err.	z	P> z	[95% conf. interval]
hrate					
ln_population	1.558282	.528013	2.95	0.003	.5233959 2.593169
poverty	.613853	.0971825	6.32	0.000	.4233788 .8043272
_cons	-13.22932	6.081798	-2.18	0.030	-25.14943 -1.309218
W					
hrate	.0652294	.0958409	0.68	0.496	-.1226154 .2530741
var(e.hrate)	34.34834	3.852857			27.56934 42.79422

Wald test of spatial terms: chi2(1) = 0.46 Prob > chi2 = 0.4961

The coefficients of explanatory variables are very similar to those from GS2SLS estimation, while the spatial lag coefficient estimate has increased from 0.038 to 0.065 but is still small and statistically insignificant. The ML standard errors are approximately 20% smaller.

26.7.3 Prediction and marginal effects

In the AR(1) time series model $y_t = \rho y_{t-1} + \beta x_t + u_t$, distinction is made between the initial impact of a one-unit change in x_t , which equals β , and changes in subsequent periods of $\rho\beta, \rho^2\beta, \dots$ that lead to total impact of $\beta/(1 - \rho)$ if $|\rho| < 1$.

A qualitatively similar distinction is made in spatial models, where changes in the value of a regressor for any given observation spills over to potentially affect the value of the dependent variable for all observations, not just the current observation.

Prediction

The reduced form in the SAR in mean model is

$$E(\mathbf{y}|\mathbf{X}) = (\mathbf{I} - \lambda \mathbf{W})^{-1} \mathbf{X} \boldsymbol{\beta}$$

This is called the reduced-form mean for \mathbf{y} .

Defining the $N \times N$ matrix $\mathbf{S} = (\mathbf{I} - \lambda \mathbf{W})^{-1}$, it follows that $E(y_i|\mathbf{X}) = \sum_{j=1}^N S_{ij} \mathbf{x}'_j \boldsymbol{\beta}$. The reduced-form mean for the i th observation therefore depends directly on its own regressor value \mathbf{x}_i through $S_{ii} \mathbf{x}'_i \boldsymbol{\beta}$ and indirectly through regressors for all other observations through $\sum_{j=1; j \neq i}^N S_{ij} \mathbf{x}'_j \boldsymbol{\beta}$. Stacking all observations, the direct mean for \mathbf{y} is $\text{diag}(\mathbf{S}) \mathbf{X} \boldsymbol{\beta}$, and the indirect mean for \mathbf{y} is $\{\mathbf{S} - \text{diag}(\mathbf{S})\} \mathbf{X} \boldsymbol{\beta}$.

The `predict` postestimation command provides sample analogues of these means. Define $\hat{\mathbf{S}} = (\mathbf{I} - \hat{\lambda} \mathbf{W})^{-1}$. The default `rform` option gives the reduced-form prediction $\hat{\mathbf{y}} = \hat{\mathbf{S}} \mathbf{X} \hat{\boldsymbol{\beta}}$. The `direct` option gives the direct prediction $\hat{\mathbf{y}} = \text{diag}(\hat{\mathbf{S}}) \mathbf{X} \hat{\boldsymbol{\beta}}$. The `indirect` option gives the indirect prediction $\hat{\mathbf{y}} = \hat{\mathbf{S}} \mathbf{X} \hat{\boldsymbol{\beta}} - \text{diag}(\hat{\mathbf{S}}) \mathbf{X} \hat{\boldsymbol{\beta}}$. Other options of the `predict` command yield structural-form predictions.

The pseudo- R^2 measure reported in the output from `sp` estimation commands is the squared correlation between y and \hat{y} , where \hat{y} is the reduced-form mean prediction.

The `margins` postestimation command can be applied to these various predictions. For marginal effects based on reduced-form predictions, it is easier to use the postestimation command `estat impact`.

Direct, indirect, and total impacts

The reduced-form mean for the i th observation is $E(y_i|\mathbf{X}) = \sum_{j=1}^N S_{ij}\mathbf{x}'_j\boldsymbol{\beta}$. It follows that the total effect on the reduced-form mean for the i th observation of a change in the regressors for the j th observation is $\partial E(y_i|\mathbf{X})/\partial \mathbf{x}_j = S_{ij}\boldsymbol{\beta}$. For the k th regressor, we have $\partial E(y_i|\mathbf{X})/\partial x_{jk} = S_{ij}\beta_k$.

The total impact of a change in the k th regressor on the reduced-form mean is computed by averaging $\partial E(y_i|\mathbf{X})/\partial x_{jk}$ over dependent variables y_i that differ over observations as well as over regressors x_{jk} that differ over observations:

$$\frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \frac{\partial E(y_i|\mathbf{X})}{\partial x_{jk}} = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N S_{ij}\beta_k$$

The total impact is then decomposed into direct and indirect effects, similar to the earlier decomposition of the prediction. Then the direct effect is

$$\frac{1}{N} \sum_{i=1}^N \frac{\partial E(y_i|\mathbf{X})}{\partial x_{ik}} = \frac{1}{N} \sum_{i=1}^N S_{ii}\beta_k$$

This estimates the average own effect of a change in the k th regressor on the conditional mean of y given \mathbf{x} . The indirect effect is

$$\frac{1}{N} \sum_{i=1}^N \sum_{j=1; j \neq i}^N \frac{\partial E(y_i|\mathbf{X})}{\partial x_{jk}} = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1; j \neq i}^N S_{ij}\beta_k$$

which estimates the average cross effect or spillover effect of a change in the k th regressor on the conditional mean of y given \mathbf{x} .

The `estat impact` postestimation command calculates these three quantities by evaluating at the estimated parameters.

We have

Average impacts						Number of obs	=	159
	Delta-Method							
	dy/dx	std. err.	z	P> z	[95% conf. interval]			
direct ln_population poverty	1.582175 .6228135	.6478725 .1085008	2.44 5.74	0.015 0.000	.3123677 .4101558	2.851981 .8354711		
indirect ln_population poverty	.0550951 .0216879	.1648445 .0647816	0.33 0.33	0.738 0.738	-.2679942 -.1052818	.3781844 .1486576		
total ln_population poverty	1.63727 .6445013	.6563342 .1058766	2.49 6.09	0.013 0.000	.3508783 .436987	2.923661 .8520157		

The indirect effects in this example are quite small, and the direct and total effects are similar to each other and to the original parameter estimates. This is due to the small spatial lag coefficient because $\widehat{\mathbf{S}} = (\mathbf{I} - \widehat{\lambda}\mathbf{W})^{-1} \simeq \mathbf{I}$ when $\widehat{\lambda} \simeq 0$, so $S_{ii} \simeq 1$ and $S_{ij} \simeq 0$ for $j \neq i$.

26.7.4 SAR(1) in error model

Another commonly used model is the SAR(1) in error model that specifies

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$$

$$\mathbf{u} = \rho\mathbf{W}\mathbf{u} + \boldsymbol{\varepsilon}$$

where $\boldsymbol{\varepsilon}_i$ is independently distributed though may be heteroskedastic.

Note that in this model, OLS of \mathbf{y} on \mathbf{X} yields consistent estimates, though the usual standard errors are inconsistent and more efficient FGLS

estimation is possible.

FG2SLS estimation

The spatial lag in error model can be rewritten as $(\mathbf{I} - \rho\mathbf{W})\mathbf{u} = \varepsilon$, and hence

$$(\mathbf{I} - \rho\mathbf{W})\mathbf{y} = (\mathbf{I} - \rho\mathbf{W})\mathbf{X}\beta + \varepsilon$$

It follows that if ε_i is i.i.d., then the FGLS estimator $\hat{\beta}_{FGLS}$ can be obtained by OLS estimation of $(\mathbf{I} - \hat{\rho}\mathbf{W})\mathbf{y}$ on $(\mathbf{I} - \hat{\rho}\mathbf{W})\mathbf{X}$, where $\hat{\rho}$ is a consistent estimate of ρ .

The `spregress` command obtains an initial consistent estimate of ρ by solving for ρ in the equations $\hat{\varepsilon}'\mathbf{W}\hat{\varepsilon} = 0$ and $\hat{\varepsilon}'\{\mathbf{W}'\mathbf{W} - \text{diag}(\mathbf{W}'\mathbf{W})\}\hat{\varepsilon} = 0$, where $\hat{\varepsilon} = (\mathbf{I} - \rho\mathbf{W})\hat{\mathbf{u}}$ and $\hat{\mathbf{u}}$ are OLS residuals. Given $\hat{\beta}_{FGLS}$ computed using this initial estimate of ρ , the `spregress` command then obtains a more efficient second-stage estimate of ρ , one that varies according to whether the assumptions are relaxed to allow the underlying errors ε_i to be heteroskedastic.

We have

```

. * SAR(1) in error: errors independent
. spregress hrate ln_population poverty, gs2sls errorlag(W) heteroskedastic
  (159 observations)
  (159 observations (places) used)
  (weighting matrix defines 159 places)

```

Estimating rho using 2SLS residuals:

```

initial:      GMM criterion = 1.5942068
alternative:  GMM criterion = 16.262889
rescale:      GMM criterion = .04927426
Iteration 0:  GMM criterion = .04927426
Iteration 1:  GMM criterion = .01480645
Iteration 2:  GMM criterion = .01480552
Iteration 3:  GMM criterion = .01480552

```

Estimating rho using GS2SLS residuals:

```

Iteration 0:  GMM criterion = .00033792
Iteration 1:  GMM criterion = .00032573
Iteration 2:  GMM criterion = .00032573

```

Spatial autoregressive model	Number of obs = 159
GS2SLS estimates	Wald chi2(2) = 40.67
	Prob > chi2 = 0.0000
	Pseudo R2 = 0.2324

hrate	Coefficient	Std. err.	z	P> z	[95% conf. interval]
hrate					
ln_population	1.54974	.6363541	2.44	0.015	.3025091 2.796971
poverty	.638827	.1038867	6.15	0.000	.4352129 .8424411
_cons	-12.81266	7.419306	-1.73	0.084	-27.35423 1.728912
W					
e.hrate	.1146457	.1404569	0.82	0.414	-.1606448 .3899361

Wald test of spatial terms: chi2(1) = 0.67 Prob > chi2 = 0.4144

The spatial lag coefficient is small and statistically insignificant. Thus, the regression coefficients and standard errors are similar to those for the OLS estimator.

26.7.5 SARAR(1,1) model

The more general SARAR(1,1) model allows for spatial dependence through the dependent variable, errors, and regressors. Then,

$$\begin{aligned}\mathbf{y} &= \lambda \mathbf{W}\mathbf{y} + \mathbf{X}\boldsymbol{\beta} + \gamma \mathbf{V}\mathbf{X}_p + \mathbf{u} = \mathbf{Z}\boldsymbol{\delta} + \mathbf{u} \\ \mathbf{u} &= \rho \mathbf{M}\mathbf{u} + \boldsymbol{\varepsilon}\end{aligned}$$

where ε_i is independent. Here \mathbf{X}_p denotes p regressors that need not necessarily be in \mathbf{X} . The spatial weighting matrices \mathbf{W} , \mathbf{V} , and \mathbf{M} need not be distinct. For example, we may have $\mathbf{W} = \mathbf{V} = \mathbf{M}$.

With this terminology, the SAR(1) in mean model is an SARAR(1,0) model, and the SAR(1) in error model is an SARAR(0,1) model.

Consistent estimates can be obtained by IV or 2SLS estimation. Combine the exogenous regressors and the spatial lag into $\mathbf{X}^* = [\mathbf{X}, \mathbf{V}\mathbf{X}_p]$. Then the instruments for \mathbf{X}^* and $\mathbf{W}\mathbf{y}$ are the linearly independent columns of $\mathbf{H} = [\mathbf{X}^*, \mathbf{W}\mathbf{X}^*, \dots, \mathbf{W}^q\mathbf{X}^*]$. The spregress default is $q = 2$. Then $\hat{\boldsymbol{\delta}}_{2SLS} = (\hat{\mathbf{Z}}'\hat{\mathbf{Z}})^{-1}\hat{\mathbf{Z}}'\mathbf{y}$, where $\hat{\mathbf{Z}} = \mathbf{H}(\mathbf{H}'\mathbf{H})^{-1}\mathbf{H}'\mathbf{Z}$.

This 2SLS estimator is consistent but is inefficient unless errors are i.i.d. and there is no spatial error component. The spatial lag in error implies $(\mathbf{I} - \rho\mathbf{M})\mathbf{u} = \boldsymbol{\varepsilon}$, and hence $(\mathbf{I} - \rho\mathbf{M})\mathbf{y} = (\mathbf{I} - \mathbf{M})\mathbf{Z}\boldsymbol{\delta} + \boldsymbol{\varepsilon}$. Given a consistent estimate $\hat{\rho}$ of ρ , the GS2SLS estimator is $\hat{\boldsymbol{\delta}}_{GS2SLS} = (\hat{\mathbf{Z}}^{*\prime}\hat{\mathbf{Z}}^*)^{-1}\hat{\mathbf{Z}}^{*\prime}\mathbf{y}^*$, where $\mathbf{y}^* = (\mathbf{I} - \hat{\rho}\mathbf{M})\mathbf{y}$, $\mathbf{Z}^* = (\mathbf{I} - \hat{\rho}\mathbf{M})\mathbf{Z}$, $\hat{\mathbf{Z}}^* = \mathbf{H}^*(\mathbf{H}^{*\prime}\mathbf{H}^*)^{-1}\mathbf{H}^{*\prime}\mathbf{Z}$, and \mathbf{H}^* is composed of the linearly independent columns of $[\mathbf{H}, \mathbf{M}\mathbf{H}]$.

As an example, we have spatial lags in both the mean and the error and spatial lags in the two regressors. In all cases, the same spatial weighting matrix is used. We have

```

. * SARAR(1,1) model with additionally SAR in X: gs2sls
. spregress hrate ln_population poverty, gs2sls dvarlag(W) errorlag(W)
>     ivarlag(W:ln_population poverty) heteroskedastic
(159 observations)
(159 observations (places) used)
(weighting matrix defines 159 places)

```

Estimating rho using 2SLS residuals:

```

initial:      GMM criterion = 14.664337
alternative:   GMM criterion = .05894927
rescale:      GMM criterion = .05894927
Iteration 0:  GMM criterion = .05894927
Iteration 1:  GMM criterion = .04696624
Iteration 2:  GMM criterion = .0469661
Iteration 3:  GMM criterion = .0469661

```

Estimating rho using GS2SLS residuals:

```

Iteration 0:  GMM criterion = .00162061
Iteration 1:  GMM criterion = .00128954
Iteration 2:  GMM criterion = .00128891
Iteration 3:  GMM criterion = .00128891

```

Spatial autoregressive model
GS2SLS estimates

	Number of obs = 159
	Wald chi2(5) = 85.58
	Prob > chi2 = 0.0000
	Pseudo R2 = 0.2452

hrate	Coefficient	Std. err.	z	P> z	[95% conf. interval]
hrate					
ln_population	1.24439	.599829	2.07	0.038	.0687469 2.420033
poverty	.7042393	.1343533	5.24	0.000	.4409116 .967567
_cons	-11.68442	7.016308	-1.67	0.096	-25.43613 2.067295
W					
ln_population	.0059698	.363766	0.02	0.987	-.7069985 .7189381
poverty	-.4211137	.2491727	-1.69	0.091	-.9094834 .0672559
hrate	.5942967	.4283109	1.39	0.165	-.2451773 1.433771
e.hrate	-.3940704	.3810944	-1.03	0.301	-1.141002 .3528609

Wald test of spatial terms: chi2(4) = 4.32 Prob > chi2 = 0.3644

The four spatial lag parameters are statistically insignificant at level 0.05, both individually and jointly. The slope coefficients are within roughly 20% of OLS estimates and are estimated with similar precision.

26.7.6 SARAR(p,q) model

For time-series data, a flexible model is the autoregressive moving average model of orders p and q (an ARMA(p, q) model) with

$$y_t = \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p} + \mathbf{x}'_t \boldsymbol{\beta} + u_t$$

$$u_t = \rho_1 u_{t-1} + \cdots + \rho_q u_{t-q} + \varepsilon_t$$

Similarly, the SARAR(1,1) model can be generalized to an SARAR(p, q) model with

$$\mathbf{y} = \lambda_1 \mathbf{W}_1 \mathbf{y} + \cdots + \lambda_p \mathbf{W}_p \mathbf{y} + \mathbf{X} \boldsymbol{\beta} + \gamma_1 \mathbf{V}_1 \mathbf{X}_{1p} + \cdots + \gamma_r \mathbf{V}_r \mathbf{X}_{rp} + \mathbf{u}$$

$$\mathbf{u} = \rho_1 \mathbf{M}_1 \mathbf{u} + \cdots + \rho_q \mathbf{M}_q \mathbf{u} + \boldsymbol{\varepsilon}$$

where $\mathbf{W}_1, \dots, \mathbf{W}_p, \mathbf{V}_1, \dots, \mathbf{V}_r, \mathbf{M}_1, \dots, \mathbf{M}_q$ are spatial weighting matrices that need not be distinct. Such more flexible models may better capture any spatial correlations, thereby ensuring that the underlying errors ε_i are independent, an essential assumption required for estimator consistency. For example, in a geospatial setting, \mathbf{W}_1 may be a contiguity matrix, while \mathbf{W}_2 may be a distance matrix.

The `gs2s1s` option of the `spregress` command enables estimation of this more general model. The `ml` option of the `spregress` command allows only the regressors \mathbf{X} to have more than one spatial lag.

26.8 Spatial instrumental variables

The GS2SLS estimator for SAR models extends straightforwardly to models where some of the regressors are endogenous. We suppose that

$$\begin{aligned}\mathbf{y} &= \mathbf{Y}^* \boldsymbol{\pi} + \lambda \mathbf{W} \mathbf{y} + \mathbf{X} \boldsymbol{\beta} + \gamma \mathbf{V} \mathbf{X}_p + \mathbf{u} \\ \mathbf{u} &= \rho \mathbf{M} \mathbf{u} + \boldsymbol{\varepsilon}\end{aligned}$$

where we have added the $N \times p$ matrix \mathbf{Y}^* of p endogenous regressors. Note that no spatial lag is applied to these endogenous regressors.

We assume that there are $q \geq p$ instruments \mathbf{X}_{inst} available for \mathbf{Y}^* . The preceding results carry through. We again define $\mathbf{y} = \mathbf{Z} \boldsymbol{\delta} + \mathbf{u}$, where \mathbf{Z} additionally includes \mathbf{Y}^* and $\boldsymbol{\delta}$ additionally includes $\boldsymbol{\pi}$. The matrix \mathbf{H} is formed from the linearly independent columns of a matrix that additionally includes $\mathbf{X}_{\text{inst}}, \mathbf{W} \mathbf{X}_{\text{inst}}, \dots, \mathbf{W}^q \mathbf{X}_{\text{inst}}$. The estimator can be implemented using the `spivregress` command.

As a purely numerical illustrative example, we consider the SAR(1) in mean model and assume that `poverty` is endogenous and can be instrumented by `gini` (which requires the identifying assumption that `gini`, the Gini coefficient of family inequality, should not be included in the original model). The instrument is very strong: the sample correlation between `poverty` and `gini` is 0.88. We expect little loss of efficiency in this example, and we expect OLS and IV estimates to be similar. We obtain

```

. * Endogenous regressors in SAR in mean model
. spivregress hrate ln_population (poverty=gini), gs2sls dvarlag(W)
> heteroskedastic
(159 observations)
(159 observations (places) used)
(weighting matrix defines 159 places)

Spatial autoregressive model                               Number of obs =    159
GS2SLS estimates                                         Wald chi2(3)   =  62.56
                                                               Prob > chi2   = 0.0000
                                                               Pseudo R2     = 0.2323


```

hrate	Coefficient	Std. err.	z	P> z	[95% conf. interval]
hrate					
poverty	.7273454	.1056205	6.89	0.000	.520333 .9343579
ln_population	1.904222	.6319827	3.01	0.003	.6655591 3.142886
_cons	-17.53529	7.172499	-2.44	0.014	-31.59313 -3.477449
W					
hrate	-.0118891	.1152612	-0.10	0.918	-.2377969 .2140186

Wald test of spatial terms: chi2(1) = 0.01 Prob > chi2 = 0.9178
 Instrumented: poverty (W*hrate)
 Raw instruments: ln_population gini hrate:_cons

As expected, the IV estimates and standard errors are similar to those from OLS regression.

26.9 Spatial panel-data models

A range of models and estimators has been proposed for spatial data observed over several time periods. The `spxtregress` command enables estimation of one of these models.

Specifically, the `spxtregress` command extends the cross-sectional SARAR model to panel data in the case where 1) the panel is strongly balanced; 2) the same spatial weighting matrix is used in each time period; 3) the model is static (so lagged dependent variables do not appear as a regressor); and 4) underlying errors are i.i.d. over time and across observations.

A fixed-effects estimator allows an individual-specific intercept that is potentially correlated with model errors. Two variants of a random-effects estimator allow the model error to include an individual-specific component that is normally distributed.

A panel variant of the SARAR(1,1) model specifies for the t th period

$$\begin{aligned}\mathbf{y}_t &= \lambda \mathbf{W} \mathbf{y}_t + \mathbf{X}_t \boldsymbol{\beta} + \boldsymbol{\alpha} + \mathbf{u}_t, \quad t = 1, \dots, T \\ \mathbf{u}_t &= \rho \mathbf{M} \mathbf{u}_t + \boldsymbol{\varepsilon}_t\end{aligned}$$

where using more compact notation \mathbf{X}_t is an $N \times K$ matrix that includes any original regressors plus any relevant spatial lags. Here \mathbf{y}_t , \mathbf{u}_t , and $\boldsymbol{\varepsilon}_t$ are $N \times 1$ vectors, and \mathbf{W} and \mathbf{M} are $N \times N$ spatial weighting matrices that are time invariant. The errors $\boldsymbol{\varepsilon}_{it}$ are assumed to be i.i.d. over both i and t .

The key complication is the addition of an $N \times 1$ vector $\boldsymbol{\alpha}$ of individual specific time-invariant additive effects.

In a fixed-effects model, the time-invariant individual effects α_i may be correlated with regressors. Define the usual mean-differencing transformation matrix $\mathbf{Q} = \mathbf{I}_T - (1/T)\mathbf{e}'_T \mathbf{e}_T$. Premultiplying the model by \mathbf{Q} eliminates the fixed effects and leads to a transformed SARAR model, but it induces correlation in errors over time. [Lee and Yu \(2010\)](#) propose an

alternative differencing transformation without this problem. The $T \times T$ orthonormal eigenvector matrix of \mathbf{Q} can be expressed as $[\mathbf{F}(\mathbf{e}_T/\sqrt{T})]$, where \mathbf{F} is the $T \times (T - 1)$ submatrix corresponding to the eigenvalues equal to 1. Then postmultiplying \mathbf{y}_t , \mathbf{X}_t , \mathbf{u}_t , and ε_t by \mathbf{F} yields transformed model

$$\begin{aligned}\mathbf{y}_t^* &= \lambda \mathbf{W} \mathbf{y}_t^* + \mathbf{X}_t^* \boldsymbol{\beta} + \mathbf{u}_t^*, \quad t = 1, \dots, T \\ \mathbf{u}_t^* &= \rho \mathbf{M} \mathbf{u}_t^* + \varepsilon_t^*,\end{aligned}$$

with underlying errors ε_{it} that are uncorrelated for all i and t (and independent under normality).

The `spxtregress` command with `fe` option fits this model. The estimator is a quasi-ML estimator that maximizes the log likelihood under the assumption of normally distributed errors ε_{it} , but the asymptotic properties (as $N \rightarrow \infty$ with T possibly fixed) of consistency and asymptotic normality require only the weaker assumption that the errors ε_{it} are i.i.d.

The `spxtregress` command with `re` option estimates models with random effects. The default `re` option supposes α_i and ε_{it} are i.i.d. normal and obtains the MLE. The `re` and `sarpanel` options fit an alternative model where α is dropped from the structural equation for \mathbf{y}_t and instead appears in the SAR model for \mathbf{u}_t , so $\mathbf{u}_t = \rho \mathbf{M} \mathbf{u}_t + \alpha + \varepsilon_t$. Again, α_i and ε_{it} are assumed to be i.i.d. normal, and the MLE is obtained.

For further details and an application, see the Stata documentation for the command `spxtregress`.

If the only spatial complication in a panel dataset is spatial correlation in the error, rather than in the conditional mean, then the panel OLS estimator is consistent. If $N \rightarrow \infty$ and $T \rightarrow \infty$ with time-series autocorrelation disappearing after m lags, then the Newey–West-type standard errors of [Driscoll and Kraay \(1998\)](#) can be obtained using the community-contributed `xtscc` command ([Hoechle 2007](#)). This does not require specification of a model for the spatial correlation in the error.

26.10 Additional resources

A lengthy description is given in [SP] *Stata Spatial Autoregressive Models Reference Manual*. Standard references for spatial econometrics are [Anselin \(1988\)](#) and [LeSage and Pace \(2009\)](#). [Conley \(1999\)](#) and [Kelejian and Prucha \(2007\)](#) present spatial HAC standard errors following OLS and IV regression.

A good introduction to SAR models is provided in [Drukker, Prucha, and Raciborski \(2013\)](#). This illustrates how redundant community-contributed Stata commands that are quite similar to the subsequent Stata `sp` commands. For relevant theory, see especially [Drukker, Egger, and Prucha \(2019\)](#) for quite general SARAR models fit by least-squares methods and [Lee \(2004\)](#) for ML estimation. [Gibbons and Overman \(2012\)](#) provide a critique of the methods presented in this chapter.

There are many community-contributed Stata spatial commands; some but not all have been superseded by the Stata `sp` commands.

A closely related subject is network analysis. Leading textbooks include [Jackson \(2010\)](#) and [Newman \(2018\)](#). [de Paula \(2020\)](#), [Graham \(2020\)](#), and [Graham and de Paula \(2020\)](#) provide recent surveys. [Hedström and Grund \(Forthcoming\)](#) detail methods using their `nwcommands` package.

26.11 Exercises

1. Open the files `mus226columbusdb.dta` and `mus226columbuscoord.dta`, and provide a brief summary of their contents. Next, `spset` the data using the following commands:
`usemus226columbusdb.dta, clear` and `spsetID` and `spset, modify shpfile(mus226columbuscoord.dta)`. Finally, give `grmap CRIME`. Do residential burglaries and vehicle thefts appear to be spatially correlated?
2. Continue with the same dataset as the previous question. Create a spatial weighting matrix (named `w`) that is a contiguity weighting matrix with default spectral normalization. Perform OLS regression of `CRIME` on household income and housing value. Comment on the fitted relationship. Perform the Moran test. Are the model errors spatially uncorrelated? Test at level 0.05. Estimate an SAR(1) in mean model. Are there spillover effects? Use command `estat impact`. Estimate an SAR(1) in error model, and compare estimates with those from OLS and SAR(1) in mean models.
3. The following creates a heat map for median age across U.S. states. The datasets available at the Stata Press website are `State_2010Census_DP1.shp` (a shape file), `State_2010Census_DP1.dbf` (a datafile), and `DP_TableDescriptions.xls` (which describes the variables in the `.dbf` file). Convert the shapefile to Stata format with command
`shp2dta` using `State_2010Census_DP1`, database (`usdb`) coordinates (`uscoord`) `genid(ID)` `replace`, which will place the coordinates in `uscoord.dta` and the dataset in `usdb.dta` with `ID`. Open file `uscoord.dta`, and summarize and list the first 10 observations. What do you learn? Open file `usdb.dta`, give command `list ID NAME`, and summarize variable `DP0020001`, which is median age in each state. What do you learn? Next, `spset` the data using commands `spset ID` and `spset, modify shpfile(uscoord.dta)`. Finally, produce a heat map with the outlying states dropped using command `grmap DP0020001 if ID != 8 & ID != 18 & ID != 35`. What do you conclude about the age distribution in the United States?

4. A peer-effects model includes the average value of variables (y or x , or both) as regressors. We use `mus206vlss.dta` with dependent variable `pharvis`, the number of pharmacy visits by an individual. Individuals are in communes, the peer group, with on average 148 people per commune. Throughout, use cluster-robust standard errors that cluster on commune. Create the variable `pharvispeer` using commands `bysort commune: egen numincommune = count(_n)` followed by `by commune: egen sumpharviscommune = sum(pharvis)` and finally `generate pharvispeer = (sumpharviscommune - pharvis) / (numincommune - 1)`. Fit an OLS regression of `pharvis` on `lnhhexp` and `illness` using cluster-robust standard errors. Provide interpretations of the coefficients of `lnhhexp` and `illness`. Now, add `pharvispeer` as a regressor. Does the average number of pharmacy visits by others in the commune appear to be related to one's own pharmacy visits? One possibility is that illnesses are socially contagious. Create variable `illnesspeer` for the average number of illnesses of others in the commune. Rerun the initial regression with this variable added as a regressor. Does illness appear to be socially contagious? Suppose we believe `illnesspeer` does not directly affect `pharvis`. Then this is available as an instrument for `pharvispeer`. Fit a regression of `pharvis` on `lnhhexp`, `illness`, and `pharvispeer` with `illnesspeer` an instrument for `pharvispeer`.
5. A better model for the data of the previous question is a Poisson model because the dependent variable `pharvis` is a count. Repeat the same regressions, except use `poisson` rather than `regress` and use `gmm` rather than `ivregress`, based on the nonlinear moment condition

$$E[\mathbf{z}_i \{y_i - \exp(\mathbf{x}'\boldsymbol{\beta})\}] = \mathbf{0}.$$

Chapter 27

Semiparametric regression

27.1 Introduction

In this chapter, we present regression methods that require less model specification than the methods of the preceding chapters.

We begin with nonparametric regression of y on \mathbf{x} , with conditional mean $E(y|\mathbf{x}) = m(\mathbf{x})$, where the functional form for $m(\cdot)$ is not specified. The most commonly used nonparametric method is kernel-weighted local polynomial regression, most often local constant or local linear. An introductory treatment was given in sections 2.6.6 and 14.6. For scalar regressor x , the `lpoly` command implements these methods with a fixed bandwidth with default value determined by a plugin formula, and the `lowess` command provides a variation to local linear and local constant regression that uses a variable bandwidth and downweights observations with large residuals.

In this chapter, we consider extension to multiple regressors. The `npregress kernel` command implements kernel-weighted local linear and local constant regression. For local linear regression, it also provides estimates of $m'(\mathbf{x}) = \partial E(y|\mathbf{x})/\partial \mathbf{x}$, enabling estimation of marginal effects. The bandwidths for the kernel are determined by leave-one-out cross classification. The `npregress series` command implements regression on higher-order polynomials or on splines.

A major limitation of nonparametric regression is the curse of dimensionality. As the number of regressors grows, the fraction of the sample available to evaluate $m(\mathbf{x})$ diminishes. For example, if with one regressor x we are willing to evaluate $m(x)$ by averaging y over values of x that fall in to 1 of 10 bins, then with 2 regressors, x_1 and x_2 , we evaluate $m(x_1, x_2)$ by averaging y over values that fall into 1 of $10^2 = 100$ bins. Theoretically, the rate of convergence of $\hat{m}(\mathbf{x})$ to $m(\mathbf{x})$ declines, and in practice the estimates $\hat{m}(\mathbf{x})$ are too noisy unless there are many observations.

The `npregress kernel` and `npregress series` commands provide additional commands that help overcome this limitation. For example, after

obtaining $\widehat{m}(x_1, x_2)$, one can compute $\widehat{m}(x_1|x_2 = x_2^*) = \widehat{m}(x_1, x_2^*)$ or $\widehat{m}(x_1) = (1/N) \sum_{i=1}^N \widehat{m}(x_1, x_{2i})$. These quantities are more precisely estimated and can be easily plotted against x_1 . Stata provides commands to do this. Related postestimation commands additionally provide ways to interpret how the dependent variable changes as regressor values change.

An alternative is to use semiparametric methods that introduce a parametric component that greatly reduces the dimension of the nonparametric component, often to dimension one. Three leading examples are the partial linear model, which specifies $E(y|\mathbf{x}, z) = \mathbf{x}'\boldsymbol{\beta} + g(z)$, where $g(\cdot)$ is not specified; the single-index model, which specifies $E(y|\mathbf{x}) = g(\mathbf{x}'\boldsymbol{\beta})$, where $g(\cdot)$ is not specified; and the generalized additive model (GAM), where $E(y|\mathbf{x}) = g_1(x_1) + \dots + g_k(x_k)$. In the first two models, the goal is to obtain a root- N consistent and asymptotically normal estimator for the parameters $\boldsymbol{\beta}$. Any nonparametric components are also consistently estimated but with rate of convergence less than root- N .

The chapter begins with details on the methods for kernel-weighted local linear and local constant regression and for series regression. The subsequent two sections present examples for regression with, respectively, a single regressor and multiple regressors. The remaining sections present various semiparametric models and estimators.

The machine learning literature has introduced additional flexible methods for regression of y on \mathbf{x} , such as lasso, neural networks, regression trees, and random forests. These methods are presented in chapter [28](#).

27.2 Kernel regression

We present kernel-weighted local linear and local constant regression and details on kernel functions and on bandwidth choice. The discussion is more expansive than the introductory treatment in sections 2.6.6 and 14.6.

27.2.1 Kernel-weighted local constant regression

Consider the regression model $y = m(\mathbf{x}) + u$, where there are K regressors and $E(u|\mathbf{x}) = 0$, so $E(y|\mathbf{x}) = m(\mathbf{x})$. The goal is to estimate the conditional mean $m(\mathbf{x})$ at each sample value of \mathbf{x} , without placing any restrictions on the functional form for $m(\cdot)$.

Suppose there are many observations on y for a given value \mathbf{x}_0 of the regressors. Then an obvious nonparametric estimator of $m(\mathbf{x}_0)$ is the average value of y for those observations with $\mathbf{x}_i = \mathbf{x}_0$. This can be expressed as $\hat{m}(\mathbf{x}_0) = \sum_{i=1}^N \{\mathbf{1}(\mathbf{x}_i = \mathbf{x}_0)/N_0\} \times y_i$, where the indicator function $\mathbf{1}(A) = 1$ if event A occurs and equals 0 otherwise and N_0 is the number of observations with $\mathbf{x}_i = \mathbf{x}_0$.

In practice, there will generally be few to no observations with $\mathbf{x}_i = \mathbf{x}_0$, especially if there are several regressors or continuous regressors, or both. A more general method additionally averages y over neighboring observations that have \mathbf{x}_i close to \mathbf{x}_0 , with greatest weight given to those observations with \mathbf{x}_i closest to \mathbf{x}_0 .

Kernel regression, also known as local constant regression, uses the locally weighted average

$$\hat{m}(\mathbf{x}_0) = \sum_{i=1}^N w(\mathbf{x}_i, \mathbf{x}_0, \mathbf{h}) \times y_i$$

where the weights $w(\mathbf{x}_i, \mathbf{x}_0, \mathbf{h})$, defined after (27.1) below, increase as \mathbf{x}_i becomes closer to \mathbf{x}_0 and decrease as the bandwidth parameter or smoothing parameter \mathbf{h} increases.

For multivariate regressor \mathbf{x} with K regressors $x_j, j = 1, \dots, K$, Stata uses the product weighting function or product kernel function

$$w(\mathbf{x}_i, \mathbf{x}_0, \mathbf{h}) = \prod_{j=1}^K w_j(\mathbf{x}_i, \mathbf{x}_0, h_j) \quad (27.1)$$

where for the j th regressor the weights are kernel weights

$$w_j(\mathbf{x}_i, \mathbf{x}_0, h) = K_j \left(\frac{x_{ji} - x_{j0}}{h} \right) \Bigg/ \sum_{i=1}^N K_j \left(\frac{x_{ji} - x_{j0}}{h} \right)$$

Kernel functions for continuous regressors were introduced in section 2.6.4 and are defined in [R] **kdensity**. For discrete regressors, the Li–Racine kernels defined in (27.2) and (27.3) below are used.

27.2.2 Kernel-weighted local linear regression

The local constant regression estimator simply averages y over \mathbf{x} values in the local neighborhood of \mathbf{x}_0 . For \mathbf{x}_0 close to the boundary of the range of \mathbf{x} , one can average only on one side of \mathbf{x}_0 , leading to bias if the relationship between y and \mathbf{x} is not constant at the boundary. Local linear regression reduces this problem by allowing for linearly increasing or decreasing relationships near the boundary.

Recall that ordinary least-squares (OLS) regression on only an intercept gives fitted value of the sample mean, and weighted least-squares regression on only an intercept gives fitted value of the weighted mean.

Thus, the local constant estimator of $\hat{m}(\mathbf{x}_0)$ can be obtained as $\hat{m}(\mathbf{x}_0) = \hat{\alpha}_0$, where $\hat{\alpha}_0$ minimizes

$$\sum_{i=1}^N w(\mathbf{x}_i, \mathbf{x}_0, \mathbf{h}) \times (y_i - \alpha_0)^2$$

The local linear estimator at $\mathbf{x} = \mathbf{x}_0$ instead minimizes with respect to α_0 and β_0 the weighted sum of squares

$$\sum_{i=1}^N w(\mathbf{x}_i, \mathbf{x}_0, \mathbf{h}) \times \{y_i - \alpha_0 - (\mathbf{x}_i - \mathbf{x}_0)' \boldsymbol{\beta}_0\}^2$$

Then the conditional mean estimate is $\hat{m}(\mathbf{x}_0) = \hat{\alpha}_0$.

Furthermore, this yields the partial-effects estimate $\hat{m}'(\mathbf{x}_0) = \hat{\boldsymbol{\beta}}_0$, where $m'(\mathbf{x}_0) = \partial m(\mathbf{x}) / \partial \mathbf{x}|_{\mathbf{x}_0}$. These estimators are consistent for $m(\mathbf{x}_0)$ and $m'(\mathbf{x}_0)$ as the sample size $N \rightarrow \infty$ and the bandwidth $h \rightarrow 0$ at an appropriate rate.

Note that for some values of \mathbf{x}_0 , the local linear estimate $\hat{m}(\mathbf{x}_0)$ may not exist, and $m(\mathbf{x}_0)$ is therefore not identified. Consider regression on a single continuous regressor. To calculate $\hat{\alpha}_0$ and $\hat{\boldsymbol{\beta}}_0$ by linear regression at a particular point x_0 , we need at least two observations x_i for which the kernel weights $K\{(x_i - x_0)/h\}$ are nonzero. This may not occur in places where the data on x are widely spread and the bandwidth h is small. Such lack of identification becomes more likely as more continuous regressors are added.

27.2.3 Kernel functions

For continuous regressors, Stata offers a range of kernel functions that are defined in [R] **kdensity**. The most commonly used are the rectangular,

Epanechnikov, and Gaussian kernels.

The rectangular or uniform kernel, the `kernel(rectangle)` option, specifies that $K(z) = (1/2)\mathbf{1}(z \leq 1)$, where $\mathbf{1}(A) = 1$ if event A occurs and $\mathbf{1}(A) = 0$ otherwise. Then one simply averages over those observations with $|(\mathbf{x}_i - \mathbf{x}_0)/h| \leq 1$ or, equivalently, with $|x_i - x_0| \leq h$.

There are two Stata variants of the Epanechnikov or parabolic kernel. The option `kernel(epanechnikov)`, the default for continuous regressors, uses the original parameterization

$K(z) = (3/4\sqrt{5})(1 - (1/5)z^2) \times \mathbf{1}[z \leq \sqrt{5}]$. The `kernel(epan2)` option uses the more commonly used parameterization

$K(z) = (3/4)(1 - z^2) \times \mathbf{1}[z \leq 1]$. Using the `kernel(epanechnikov)` option with bandwidth $h = b$ is equivalent to using the `kernel(epan2)` option with bandwidth $h = \sqrt{5}b$. The Epanechnikov kernel is the optimal kernel (minimizing mean-squared error [MSE]) for density estimation but not necessarily for kernel regression.

The Gaussian or normal kernel specifies $K(z) = (1/\sqrt{2\pi})e^{-z^2/2}$. Because $K(z)$ is then positive for all z , it is less likely to lead to lack of identification of $m(\mathbf{x}_0)$ for some values of \mathbf{x}_0 .

For discrete binary regressors, the default kernel for the `npregress` command is to use the Li–Racine kernel. For scalar regressor x , this defines

$$K(x_i - x_0, h) = \begin{cases} 1 & \text{if } x_i = x_0 \\ h & \text{otherwise,} \end{cases} \quad 0 \leq h \leq 1 \quad (27.2)$$

When $h = 0$, the Li–Racine kernel reduces to the indicator function $\mathbf{1}(x_i = x_0)$. In the simplest case of nonparametric regression of y_i on binary regressor d_i , this leads to $\hat{m}_i(0)$ equal to the average of y when $d_i = 0$ and $\hat{m}_i(1)$ equal to the average of y when $d_i = 1$; the same result is obtained using the `dkernel(cellmean)` option.

When $h = 1$, the Li–Racine kernel is a uniform kernel, always equal to 1. Then nonparametric regression of y on binary regressor d leads to $\hat{m}(0) = \hat{m}(1) = \bar{y}$.

For ordered discrete regressors, the default kernel is the Li–Racine kernel. For regressor x_j taking possible values $0, \dots, m$, this defines

$$K(x_i - x_0, h) = h^{|x_{ji} - x_{j0}|}, \quad 0 \leq h \leq 1 \quad (27.3)$$

This reduces to the kernel (27.2) in the binary case. Similar to the binary case, when $h = 0$, the kernel yields cell means, equivalent to the `dkernel(cellmean)` option, and when $h = 1$, the kernel yields the overall mean. When the ordered discrete regressor takes many values, an alternative is to treat it as a continuous regressor.

27.2.4 Bandwidth choice

Crucial ingredients are the smoothing parameters h_j , $j = 1, \dots, K$, that determine the bandwidth. Smaller values of h_j lead to smaller bias in the estimate $\hat{m}(\mathbf{x}_0)$ because greater weight is then placed on observations with \mathbf{x}_i close to \mathbf{x}_0 . But smaller values of h_j also lead to greater variance in the estimate $\hat{m}(\mathbf{x}_0)$ because then fewer observations are used in computing $\hat{m}(\mathbf{x}_0)$.

This tradeoff is balanced by minimizing MSE, the sum of squared bias and variance. Empirically, this is implemented by choosing the bandwidths for each regressor to jointly minimize the leave-one-out cross-validation (LOOCV) measure

$$\text{LOOCV} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{m}_{-i})^2$$

where evaluation of $m(\mathbf{x}_0)$ is at each of the N sample values $\mathbf{x}_1, \dots, \mathbf{x}_N$, and \widehat{m}_{-i} is the estimate $\widehat{m}(\mathbf{x}_i)$ of $m(\mathbf{x}_i)$ obtained using all observations but the i th observation.

In addition to calculating the conditional mean estimate $\widehat{m}(\cdot)$, the `npregress kernel` command estimates partial effects, the derivatives of $m(\mathbf{x})$ with respect to \mathbf{x} . These estimates are obtained using different bandwidths, also calculated using cross-validation. The `lpoly` command instead uses a plugin value for the bandwidth.

27.2.5 The `npregress kernel` command

The `npregress kernel` command has syntax

```
npregress kernel depvar indepvars [ if ] [ in ] [ , options ]
```

The default is to use the local linear kernel estimator. The local constant kernel estimator is obtained using the `estimator(constant)` option; then the `noderivatives` option is additionally needed if regressors are continuous. The *options* include `kernel()`, which specifies the kernel function to be used for all the continuous regressors. Available kernels are defined in [R] **kdensity**; the default is `kernel(epanechnikov)`. The `dkernel()` option specifies the kernel function to be used for all the discrete regressors, which need to be identified using the `i.` prefix. The default is `dkernel(liracine)`, while `dkernel(cellmean)` uses cell means. The `vce(bootstrap)` option provides bootstrap standard errors and associated p -values and confidence intervals for the averages of $\widehat{m}(\mathbf{x}_0)$ and $\widehat{m}'(\mathbf{x}_0)$. Other options include ones that allow direct specification of bandwidths for conditional mean estimates and for partial-effect estimates. The evaluation points \mathbf{x}_0 are the distinct sample values of \mathbf{x} .

Note that the flexibility of nonparametric regression comes at the cost of increased variability and bias in finite samples, due to local weighting. The asymptotic theory assumes that the bandwidth $h \rightarrow 0$ as the sample size $N \rightarrow \infty$. With K regressors, the optimal bandwidth that minimizes MSE is $O(N^{-1/(K+4)})$, in which case $\widehat{m}(\mathbf{x}_0)$ converges to $m(\mathbf{x}_0)$ at rate

$O(N^{-2/(K+4)})$, rather than the usual $O(N^{-1/2})$ for OLS. And the bias in $\hat{m}(\mathbf{x}_0)$ is also $O(N^{-2/(K+4)})$.

The estimates $\hat{m}(\mathbf{x}_0)$ and $\hat{m}'(\mathbf{x}_0)$ are asymptotically normally distributed. The default for the `npregress` command is to present no estimates of precision. The `vce(bootstrap)` option provides bootstrap standard errors and associated p -values and confidence intervals for the averages of $\hat{m}(\mathbf{x}_0)$ and $\hat{m}'(\mathbf{x}_0)$. In presenting confidence intervals, Stata follows the standard approach of ignoring any possible bias in $\hat{m}(\mathbf{x}_0)$ and $\hat{m}'(\mathbf{x}_0)$, though in practice such bias will be nonnegligible unless the local average is over many observations.

27.2.6 Kernel-weighted local linear m-estimation

Local polynomial estimators can be generalized from least-squares estimators to m -estimators such as maximum likelihood estimators that maximize $\sum_{i=1}^N q(y_i | \mathbf{x}_i, \boldsymbol{\theta})$.

A local m -estimator at $\mathbf{x} = \mathbf{x}_0$ maximizes with respect to $\boldsymbol{\theta}_0$ the weighted sum

$$\sum_{i=1}^N w(\mathbf{x}_i, \mathbf{x}_0, \mathbf{h}) \times q\{y_i | (\mathbf{x}_i - \mathbf{x}_0), \boldsymbol{\theta}_0\}$$

where $w(\cdot)$ are kernel weights.

An example given in section [17.8.4](#) uses a local logit model to obtain nonparametric estimates of $\Pr(y_i = 1 | \mathbf{x}_i)$.

27.3 Series regression

A series estimator fits a model with regressors that are based on a series expansion in \mathbf{x} . Leading examples are polynomials in \mathbf{x} , introduced in section 14.4, and regression splines in \mathbf{x} , introduced in section 14.5.

A nonparametric series estimator allows the order of the polynomial or the number of knots, or both, to increase with sample size, in which case these quantities become the analogue of the bandwidth in kernel regression and are determined by data-driven methods such as cross-validation or information criterion.

27.3.1 Series regression model

A series expansion model or basis function model or sieve regression model specifies

$$E(y_i | \mathbf{x}_i) = \sum_{j=1}^M \beta_j g_j(\mathbf{x}_i)$$

where the M functions $g_j(\cdot)$ are series expansions of the underlying \mathbf{x}_i .

In the scalar regressor case, the $g_j(\cdot)$ may be a polynomial or regression spline in x_i . For multiple regressors, these terms are fully interacted in the case of regression splines, while for a polynomial of degree K , the sum of the powers of cross-product terms is restricted to be at most K .

A series regression model can be directly estimated by OLS regression, and appropriate standard errors are easily calculated.

27.3.2 The npregress series command

The `npregress series` command has syntax

```
npregress series depvar indepvars [if] [in] [weight] [, options]
```

For polynomial regression, the `polynomial()` option fits a polynomial of data-determined degree, while the `polynomial(#)` option fits a polynomial of specified degree.

For spline regression, the `spline` option fits a third-order natural spline, and the `spline(#)` option fits a natural spline of specified order. The command also estimates B-splines, using the `bspline` and `bspline(#)` options. For spline regression, the number of knots can be data determined or can be specified. In the latter case, the `knots(#)` option specifies the number of knots, or the `knotsmat()` option provides values for the knots.

The `criterion()` option specifies the method used to determine the polynomial degree or the number of knots, if these have not been specified. The available methods are LOOCV, generalized cross-validation, Akaike's information criterion, Bayesian information criterion, and Mallow's criterion.

The `vce()` option enables inference based on heteroskedastic–robust standard errors (the default), homoskedastic standard errors, or bootstrap standard errors (including cluster bootstrap standard errors).

The `asis()` option enables regressors to be included as independent regressors, rather than as polynomials or splines. The `nointeract()` option enables a polynomial or spline in a variable to appear on its own rather than interactively.

27.4 Nonparametric single regressor example

We use `mus202psid92m.dta`, which was introduced in section 2.4. To speed analysis, we analyze a 5% subsample, and to make graphs easier to read, we drop a few observations with annual earnings over \$145,000. We present the case of a single regressor in some detail before extension to multiple regressors.

27.4.1 Basic kernel-weighted local linear regression

Summary statistics for the level of annual earnings and various regressors (annual hours, age in years, and four education categories) in the subsample are

```
. * Read in data and choose a 5% sample
. qui use mus202psid92m
. drop if earnings==0 | earnings==. | earnings > 145000
(521 observations deleted)
. set seed 10101
. keep if runiform() < 0.05
(3,588 observations deleted)
. sum earnings hours age edcat ibn.edcat
```

Variable	Obs	Mean	Std. dev.	Min	Max
earnings	181	29883.24	21727.37	700	132000
hours	181	2094.37	671.4728	180	3840
age	181	37.92265	5.628754	30	50
edcat	181	2.618785	1.018431	1	4
edcat	1	.1491713	.3572454	0	1
2	181	.3314917	.4720552	0	1
3	181	.2707182	.4455634	0	1
4	181	.2486188	.4334112	0	1

OLS regression of `earnings` on `hours` yields a slope coefficient of 15.14 and $R^2 = 0.219$.

```

. * Single regressor: OLS
. regress earnings hours, vce(robust)

Linear regression
Number of obs      =      181
F(1, 179)          =     32.60
Prob > F          =     0.0000
R-squared           =     0.2188
Root MSE            =    19257

```

earnings	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
hours	15.1373	2.651025	5.71	0.000	9.906017	20.36858
_cons	-1819.867	4932.261	-0.37	0.713	-11552.72	7912.99

We then obtain the local linear estimator using the `epan2` kernel.

```

. * Single regressor: Local linear with epan2 kernel and no standard errors
. npregress kernel earnings hours, kernel(epan2)

Computing mean function
Minimizing cross-validation function:
Iteration 0: Cross-validation criterion = 50.618809
Iteration 1: Cross-validation criterion = 50.618809

warning: 1 observation was not used to compute the mean function because it
violated the model identification assumptions. This observation is
marked as 1 in the system variable _unident_sample. You may use the
unidentsample() option to use a different variable name.

Computing optimal derivative bandwidth
Iteration 0: Cross-validation criterion = 51380.609
Iteration 1: Cross-validation criterion = 2899.7862
Iteration 2: Cross-validation criterion = 2899.7862
Iteration 3: Cross-validation criterion = 2899.7862
Iteration 4: Cross-validation criterion = 2899.7862
Iteration 5: Cross-validation criterion = 2899.7862
Iteration 6: Cross-validation criterion = 2544.6791
Iteration 7: Cross-validation criterion = 2544.6791
Iteration 8: Cross-validation criterion = 2328.178
Iteration 9: Cross-validation criterion = 2328.178
Iteration 10: Cross-validation criterion = 2328.178
Iteration 11: Cross-validation criterion = 2284.5741
Iteration 12: Cross-validation criterion = 2284.5741
Iteration 13: Cross-validation criterion = 2265.005
Iteration 14: Cross-validation criterion = 2265.005
Iteration 15: Cross-validation criterion = 2253.2349
Iteration 16: Cross-validation criterion = 2253.2349
Iteration 17: Cross-validation criterion = 2246.9712
Iteration 18: Cross-validation criterion = 2246.9712
Iteration 19: Cross-validation criterion = 2246.9712
Iteration 20: Cross-validation criterion = 2245.4269
Iteration 21: Cross-validation criterion = 2245.4269

```

Bandwidth

	Mean	Effect
hours	237.4066	1149.882

Local-linear regression	Number of obs	=	180
Kernel : epan2	E(Kernel obs)	=	180
Bandwidth: cross-validation	R-squared	=	0.2676

earnings	Estimate
Mean earnings	29869.77
Effect hours	14.27923

Note: Effect estimates are averages of derivatives.

Note: You may compute standard errors using vce(bootstrap) or reps().

The initial output reports the iterative decline in the LOOCV criterion as the bandwidth is changed to ultimately yield the optimal bandwidth for $\hat{m}(\mathbf{x})$ and for $\hat{m}'(\mathbf{x})$. One observation needed to be dropped; this observation is identified below. The table then reports the optimal bandwidths. The first bandwidth, the mean bandwidth, is the one that minimizes LOOCV for prediction of the conditional mean $m(\mathbf{x})$. The second bandwidth is used for estimating the partial effects $m'(\mathbf{x})$ and is wider because the rate of convergence for $\hat{m}'(\mathbf{x})$ is slower than that for $\hat{m}(\mathbf{x})$. In this example, it is exceptionally wide because $2h \simeq 2300$ is more than 3 times the standard deviation of `hours`.

The key results are then given. The nonparametric estimates lead to $R^2 = 0.268$, a much better fit than OLS, which had $R^2 = 0.219$ (albeit fit on all 181 observations). The average predicted value of the fitted mean, $(1/N) \sum_{i=1}^N \hat{m}(x_i)$, is 29869.77, so on average, predicted earnings are \$29,870. The average of the partial effect, $(1/N) \sum_{i=1}^N \hat{m}'(x_i)$, is 14.28, so on average over the sample, an additional hour of work is associated with a \$14.28 increase in earnings. This is close to the OLS estimate (fit on all 181 observations) of 15.13.

To obtain measures of the precision of the estimates, we need to use the option `vce(bootstrap)`. We obtain

```
. * Single regressor: Local linear with epan2 kernel and bootstrap standard errors
. npregress kernel earnings hours, kernel(epan2)
> vce(bootstrap, seed(10101) reps(400) nodots)
```

Bandwidth

	Mean	Effect
hours	237.4066	1149.882

```
Local-linear regression
Kernel : epan2
Number of obs = 180
Bandwidth: cross-validation
E(Kernel obs) = 180
R-squared = 0.2676
```

	Observed estimate	Bootstrap std. err.	z	P> z	Percentile [95% conf. interval]
Mean earnings	29869.77	1544.925	19.33	0.000	26881.36 32934.34
Effect hours	14.27923	3.021378	4.73	0.000	8.151543 19.69207

Note: Effect estimates are averages of derivatives.

Note that the output no longer includes a log of the iterations to determine the optimal bandwidth. Now, additionally, we find that the average partial effect of 14.28 has standard error 3.02 and is highly statistically significant with $t = 4.73$. The flexibility of local linear regression comes at the expense of precision. Thus, the standard error for the average partial effect of hours is 3.02 compared with a standard error of 2.65 for the OLS slope coefficient (using all 181 observations).

Determining the optimal bandwidths can require considerable computation. These optimal bandwidths are held constant throughout the bootstrap replications, greatly speeding up computation that nonetheless can take considerable time. The default bootstrap is a pairs bootstrap. The option `vce(bootstrap, cluster(varlist))` implements a cluster pairs bootstrap that gives cluster-robust standard errors with clustering on the variable given in `varlist`.

27.4.2 Observations not identified

One observation was dropped because of inability to fit the model, whereas OLS can be fit using all observations. In general, observations are more likely

to be dropped the smaller the sample size and the greater the number of regressors.

One can do analysis with this observation dropped, but in general there is a reluctance to drop observations. And for some purposes, predictions for all observations may be needed.

To obtain predictions for the full sample, one can proceed in several ways. First, a wider bandwidth can be used, one wide enough that no observations are dropped. Second, an alternative kernel may be used. In particular, the Gaussian kernel sets $K(z) > 0$ for all z , whereas the other kernels set $K(z) = 0$ for larger values of z . Third, one can use the local constant estimator rather than the default local linear estimator.

In the current example, the bandwidth for $\hat{m}(x)$ is 237.4, so a local linear estimate is not possible for any observation on hours for which there is no other observation within 237.4 hours. We first identify that observation.

```
. * The nonidentified observation
. qui npregress kernel earnings hours, kernel(epan2)
. list earnings hours if _unident_sample == 1, clean
    earnings      hours
 72.      90500     3840.00
. list hours if hours > 3590, clean
    hours
 72.     3840.00
148.     3598.00
```

The problem observation is that with $\text{hours} = 3840$. The closest observation has $\text{hours} = 3598$, and $3840 - 3598 = 242 > 237.4$.

OLS estimates with this observation dropped can be compared directly with the preceding nonparametric estimates.

```

. * OLS with the nonidentified obervation dropped
. regress earnings hours if _unident_sample == 0, vce(robust)

Linear regression
Number of obs      =      180
F(1, 178)          =     29.18
Prob > F           =     0.0000
R-squared           =     0.1984
Root MSE            =    19132

```

earnings	Robust					
	Coefficient	std. err.	t	P> t	[95% conf. interval]	
hours	14.36874	2.659945	5.40	0.000	9.119659	19.61783
_cons	-407.6364	4934.983	-0.08	0.934	-10146.24	9330.964

The nonparametric estimates fit better with $R^2 = 0.268$ compared with $R^2 = 0.198$ for OLS. The average of the partial effect, $(1/N) \sum_{i=1}^N \hat{m}'(x_i)$, is 14.28, compared with the OLS slope estimate of 14.37. The standard error for the average partial effect of hours is 3.02 compared with a standard error of 2.66 for the OLS slope coefficient.

The following three variations of the `npregress` command lead to identification for all observations.

```

. * Three ways to obtain identification for all observations
. qui npregress kernel earnings hours, kernel(epan2) meanbw(243, copy)
. matrix list e(b) // Wider bandwidth for same kernel
e(b)[1,2]
      Mean:      Effect:
      earnings      hours
y1  30211.289   15.25212
. qui npregress kernel earnings hours, kernel(gaussian)
. matrix list e(b) // Different unbounded kernel
e(b)[1,2]
      Mean:      Effect:
      earnings      hours
y1  30168.583   15.895576
. qui npregress kernel earnings hours, kernel(epan2) estimator(constant)
>      noderivatives // Local constant rather than local linear
. matrix list e(b)
symmetric e(b)[1,1]
      Mean:
      earnings
y1  29681.41

```

The dropped observation had relatively high earnings, so using all 181 observations leads to a higher average fitted mean for the two local linear estimations. The local constant estimated average fitted mean of 29681 is lower because it does not allow for an increasing relationship in `hours` at high values of `hours`. And for local constant regression with a continuous regressor, partial effects are no longer available.

27.4.3 Trimmed means

Note that even if $\hat{m}(x)$ is identified for all observations, it may be very noisily estimated in regions where the data on x are sparse, leading to imprecision of the average mean $(1/N) \sum_{i=1}^N \hat{m}(x_i)$.

A common procedure is to compute a trimmed mean that drops observations for which $\hat{f}(x)$, the estimated density of x , is small. The following code computes the trimmed mean, dropping the 5% of observations with the smallest $\hat{f}(x)$, where $\hat{f}(x)$ for `hours` is calculated using the `kdensity` command. Here the kernel and bandwidth are specified to be the same as those used by the `npregress` command.

```

. * Trimmed mean dropping 5% of observations with lowest f_hat(x)
. qui npregress kernel earnings hours, kernel(epan2) meanbw(243, copy)
. qui predict m_earnings
. kdensity hours, kernel(epan2) bwidth(243) at(hours) generate(hours_x hours_d)
. qui centile hours_d, centile(5)
. list hours_d hours earnings m_earnings if hours_d < r(c_1), clean
    hours_d      hours      earnings      m_earnings
38.   .00005703    746.00       8000     6870.002
62.   .00007799    180.00       729     1375.756
72.   .00001719   3840.00      90500     90500
99.   .0000797     193.00       989     2233.159
105.   .00008132    252.00      15000     5635.232
110.   .00003947    520.00      4696     4430.957
117.   .00007686    315.00      4832     5764.087
118.   .00004466    600.00      3500     4838.011
168.   .00008169    220.00       700     3875.243
. sum m_earnings if hours_d >= r(c_1)
      Variable |      Obs       Mean     Std. dev.       Min       Max
m_earnings |      172    31062.33     9361.985    7785.32   69356.19
. sum m_earnings
      Variable |      Obs       Mean     Std. dev.       Min       Max
m_earnings |      181    30211.29     11573.35    1375.756     90500

```

All but one of the observations dropped in computing the trimmed mean have low `hours` and associated low `earnings`. Thus, the trimmed mean fitted earnings (31,062) are higher than the untrimmed mean (30,211).

The original `npregress` estimates that dropped the observation with `hours=3840` can also be seen as a form of trimming; this observation has the lowest value (0.00001719) for the fitted density.

27.4.4 Conditional means and partial effects for different regressor values

Rather than sample averages of estimated conditional means and partial effects, interest often lies in estimated conditional means and partial effects for specific values of the regressor.

The following code uses the `margins` postestimation command to calculate the fitted mean $\hat{m}(x_0)$ for `hours` equal to 1,000, 2,000, and 3,000,

along with 95% confidence intervals.

```
. * Compute predicted mean at hours = 1000, 2000, 3000
. qui npregress kernel earnings hours, kernel(epan2)
>   vce(bootstrap, seed(10101) reps(400))
. margins, at(hours=(1000(1000)3000)) vce(bootstrap, seed(10101) reps(400) nodots)
Adjusted predictions                                         Number of obs = 180
                                                               Replications = 400
Expression: Mean function, predict()
1._at: hours = 1000
2._at: hours = 2000
3._at: hours = 3000
```

	Observed margin	Bootstrap std. err.	z	P> z	Percentile [95% conf. interval]	
_at						
1	9332.563	2394.826	3.90	0.000	3784.188	12254.2
2	28084.98	1820.559	15.43	0.000	24793.01	32087.81
3	41566.51	12190.7	3.41	0.001	30744.03	54574.63

For example, the predicted conditional mean earnings at 2,000 hours is \$28,085, with 95% confidence interval [24793, 32088]. The estimates are much more precise at 2,000 hours, where the data on annual hours are relatively dense, than at 1,000 hours and 3,000 hours, where the data on hours are more sparse.

We next present marginal effects or partial effects computed as $\Delta\hat{m}(x_0)$, the change in the estimated conditional mean as x_0 changes. The `contrast(atcontrast(ar))` option of the `margins` command calculates the change in the prediction when the regressor value changes from one value to the next. The following code computes this, along with 95% confidence bands.

```

. * Compute predicted effect of change in hours of 1,000 hours
. margins, at(hours=(1000(1000)3000)) contrast(atcontrast(ar))
>     vce(bootstrap, seed(10101) reps(400) nodots)
Contrasts of predictive margins                               Number of obs = 180
                                                               Replications = 400
Expression: Mean function, predict()
1._at: hours = 1000
2._at: hours = 2000
3._at: hours = 3000

```

	df	chi2	P>chi2
_at (2 vs 1)	1	37.31	0.0000
(3 vs 2)	1	1.21	0.2712
Joint	2	39.32	0.0000

	Observed contrast	Bootstrap std. err.	Percentile [95% conf. interval]	
_at (2 vs 1)	18752.42	3069.863	14353.92	25744.2
(3 vs 2)	13481.53	12252.35	909.9065	26405.05

For example, the preceding `margins` command computed values of 9,333 at 1,000 hours and 28,085 at 2,000 hours, so a 1,000-hour increase from 1,000 hours to 2,000 hours increases earnings by $28085 - 9333 = \$18,752$. The 95% confidence interval is [14354, 25744].

27.4.5 Plots for different regressor values

The `marginsplot` command can be used to plot predicted conditional means and marginal effects for various values of the regressor.

The following code produces a plot of the relationship between estimated mean earnings and hours as hours increase in increments of 100 from 400 to 3,200, along with 95% pointwise confidence intervals and a scatterplot of the data.

```

. * Plot predictions at many levels of hours along with 95%
> confidence intervals
. qui npregress kernel earnings hours, kernel(epan2)
. qui margins, at(hours=(400(100)3200))
> vce(bootstrap, seed(10101) reps(400) nodots)
. marginsplot, legend(off)
> addplot(scatter earnings hours if earnings < 80000, msize(vsmallest))
Variables that uniquely identify margins: hours

```

The plot is given in the first panel of figure 27.1. The confidence intervals are narrower in regions where the data are denser, with more observations over a local range of hours. The relationship appears to be linear, aside from at high values of hours where the confidence intervals are very broad.

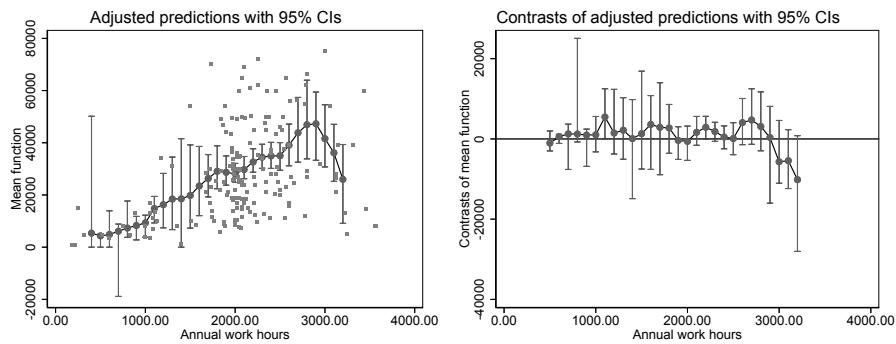


Figure 27.1. Local linear regression of earnings on hours with bootstrap confidence intervals

We next plot the corresponding marginal effects, computed as $\Delta\hat{m}(x_0)$, using the `contrast(atcontrast(ar))` option of the `margins` command, along with 95% confidence bands. We have

```

. * Plot effect of change in hours of 100 hours along with 95%
> confidence intervals
. qui margins, at(hours=(400(100)3200)) contrast(atcontrast(ar))
> vce(bootstrap, seed(10101) reps(400) nodots)
. marginsplot, yline(0)
Variables that uniquely identify margins: hours

```

The plot is given in the second panel of figure 27.1.

27.4.6 Comparison with the `lpoly` command

If interest lies only in a scatterplot of y on a single variable x with fitted nonparametric regression line, then the `lpoly` command is adequate and is much faster than the `npregress kernel` command for the following reasons. First, the `lpoly` command default is to estimate the conditional mean at 50 equally spaced data points (for $N \geq 50$), whereas the `npregress` command evaluates at each of the N observations. Second, the `lpoly` command uses a plugin estimate of the bandwidth (see [R] **Ipoly** for the method), whereas the `npregress` command uses computationally intensive cross-validation.

If additionally confidence bands are desired, the `lpoly` command is again much quicker because it does not use the bootstrap to obtain standard errors.

For a specific choice of kernel, bandwidth, and polynomial degree, the two commands give essentially the same plot of the fitted conditional mean, aside from the afore-mentioned evaluation at different values of the regressor. This is demonstrated in end-of-chapter exercise 2.

27.4.7 Basic series regression

As an example of series regression, we present a third-order regression spline with the number of knots determined using cross-validation.

```
. * Third-order natural spline with number of knots determined by
> cross-validation
. npregress series earnings hours, spline criterion(cv)
Computing approximating function
Minimizing cross-validation criterion
Iteration 0: Cross-validation criterion = 3.89e+08
Iteration 1: Cross-validation criterion = 3.81e+08
Computing average derivatives
Cubic-spline estimation                               Number of obs      =
Criterion: cross-validation                         Number of knots     =
                                                               =          181
                                                               =          3

```

	Robust					
earnings	Effect	std. err.	z	P> z	[95% conf. interval]	
hours	14.4425	4.912036	2.94	0.003	4.815081	24.06991

Note: Effect estimates are averages of derivatives.

The average marginal effect is similar to that from `npregress kernel`. Additional computations find $R^2 = 0.259$ compared with $R^2 = 0.268$ from `npregress kernel` and $R^2 = 0.198$ from OLS.

The `margins` and `marginsplot` postestimation commands that are available following the `npregress kernel` command can also be used following the `npregress series` command. For example,

```
. * Compute predicted mean at hours = 1000, 2000, 3000
. margins, at(hours=(1000(1000)3000))

Adjusted predictions                                         Number of obs = 181
Model VCE: Robust

Expression: Mean function, predict()
1._at: hours = 1000
2._at: hours = 2000
3._at: hours = 3000
```

	Delta-method					
	Margin	std. err.	z	P> z	[95% conf. interval]	
_at						
1	10439	1709.75	6.11	0.000	7087.948	13790.04
2	27626.23	1886.063	14.65	0.000	23929.62	31322.85
3	35240.05	4833.842	7.29	0.000	25765.9	44714.21

27.5 Nonparametric multiple regressor example

We consider regression of earnings on hours, age, and a set of indicator variables for educational level, specifically, less than 12 years of education, 12 years, 13–15 years, and 16 years and more.

Nonparametric local linear regression with the kernel (epan2) option yields

```
. * Multiple regressor: Local linear with epan2 kernel and bootstrap standard
> errors
. npregress kernel earnings hours age i.edcat, kernel(epan2)
> vce(bootstrap, seed(10101) reps(400) nodots)
```

Bandwidth

	Mean	Effect
hours	319.5236	454.4498
age	2.67847	3.809516
edcat	.5	.5

```
Local-linear regression
Continuous kernel : epan2
Discrete kernel   : liracine
Bandwidth        : cross-validation
Number of obs      =          164
E(Kernel obs)     =          164
R-squared          =       0.6791
```

earnings	Observed estimate	Bootstrap std. err.	Percentile		
			z	P> z	[95% conf. interval]
Mean earnings	31542.15	1955.474	16.13	0.000	27801.93 34943.35
Effect					
hours	5.489886	22.30647	0.25	0.806	-64.18196 30.25034
age	-475.8053	1142.455	-0.42	0.677	-2825.904 904.5559
edcat					
(2 vs 1)	8534.147	2522.855	3.38	0.001	2675.327 12258.37
(3 vs 1)	15411.41	5055.197	3.05	0.002	2832.935 23302.27
(4 vs 1)	22813.51	7684.48	2.97	0.003	3604.087 34439.3

Note: Effect estimates are averages of derivatives for continuous covariates and averages of contrasts for factor covariates.

```
. count if _unident_sample == 1
```

The first table gives the separate bandwidths for the two continuous regressors and the discrete ordered regressor, the bandwidths being established by cross-validation. Again, the bandwidths are larger for the partial effects than for the conditional mean. The model fits quite well with $R^2 = 0.679$. The `npregress` command with bootstrap option does not state that some observations were not identified, but because fitted means were computed for only 164 of the 181 observations, it must be that the mean was not identified for 17 observations, compared with 1 observation in the case of a single regressor.

The average partial effect of hours worked is much smaller once age and educational categories are added as regressors and is now statistically insignificant.

The most statistically significant average partial effects are those for the educational category variables. To perform a joint test of their significance, we verify the appropriate names for the stored variables using command `matrix list e(b)` and then perform the joint test.

```
. * Test joint statistical significance of education categories
. matrix list e(b)

e(b) [1,6]
      Mean:      Effect:      Effect:      Effect:      Effect:      Effect:
                  r2vs1.      r3vs1.      r4vs1.
      earnings      hours      age      edcat      edcat      edcat
y1    31542.151   5.4898856 -475.80533   8534.147  15411.408  22813.509

. test r2vs1.edcat r3vs1.edcat r4vs1.edcat
( 1) [Effect]r2vs1.edcat = 0
( 2) [Effect]r3vs1.edcat = 0
( 3) [Effect]r4vs1.edcat = 0
      chi2( 3) =    12.33
      Prob > chi2 =    0.0063
```

The educational category regressors are jointly statistically significant at 5%.

OLS estimation over the same 161 observations as identified using the preceding `npregress kernel` command yields

```

. * Multiple regressor: OLS on same sample as npregress kernel
. regress earnings hours age i.edcat if _unident_sample == 0, vce(robust)

Linear regression
                                         Number of obs      =       161
                                         F(5, 155)        =     11.24
                                         Prob > F        =     0.0000
                                         R-squared        =     0.2989
                                         Root MSE         =   18184

```

earnings	Coefficient	Robust				
		std. err.	t	P> t	[95% conf. interval]	
hours	16.60835	3.457316	4.80	0.000	9.778813	23.43789
age	146.0515	259.0136	0.56	0.574	-365.6007	657.7037
edcat						
2	10053.61	3153.741	3.19	0.002	3823.752	16283.47
3	11801.48	3204.822	3.68	0.000	5470.719	18132.25
4	19637.09	4155.196	4.73	0.000	11428.97	27845.21
_cons	-21118.26	11896.43	-1.78	0.078	-44618.3	2381.788

OLS fits worse than the local linear regression, with $R^2 = 0.300$ compared with 0.679. The OLS coefficients indicate larger average effects than local linear regression. The OLS standard errors are smaller for continuous regressors, as expected, and for the education categories aside from the second category.

Nonparametric series estimation using third-order regression splines yields

```

. * Multiple regressor: OLS on same sample as npregress kernel
. npregress series earnings hours age, spline criterion(cv) asis(i.edcat)
Computing approximating function
Minimizing cross-validation criterion
Iteration 0: Cross-validation criterion = 7.42e+08
Computing average derivatives
Cubic-spline estimation                               Number of obs      =
Criterion: cross-validation                         Number of knots     =

```

earnings	Effect	Robust				
		std. err.	z	P> z	[95% conf. interval]	
hours	13.16959	2.419944	5.44	0.000	8.426587	17.91259
	30.57134	710.2491	0.04	0.966	-1361.491	1422.634
edcat	(2 vs 1)	13047.44	3051.515	4.28	0.000	7066.586
	(3 vs 1)	14182.32	3291.746	4.31	0.000	7730.612
	(4 vs 1)	21772.14	3826.244	5.69	0.000	14272.84
						29271.45

Note: Effect estimates are averages of derivatives.

Only one knot is chosen. Additional computation yields $R^2 = 0.427$, substantially higher than OLS but lower than from kernel regression.

27.6 Partial linear model

The partial linear regression model is the regular linear regression model, except that one (or more) of the regressors appears in the model nonlinearly as $g(z)$, where the function $g(\cdot)$ is unspecified. We focus on the case of scalar z , with

$$y = \alpha + \mathbf{x}'\boldsymbol{\beta} + g(z) + u$$

The goal is to obtain a root- N consistent and asymptotically normal estimator of $\boldsymbol{\beta}$, given the usual assumption that $E(u|\mathbf{x}, z) = 0$. This provides an estimate of the marginal effect on $E(y|\mathbf{x}, z)$ of changes in \mathbf{x} .

Taking conditional expectations with respect to z , we have

$$E(y|z) = \alpha + E(\mathbf{x}|z)'\boldsymbol{\beta} + g(z)$$

given the usual assumption that $E(u|z) = 0$. Subtracting the two equations yields

$$\{y - E(y|z)\} = \{\mathbf{x} - E(\mathbf{x}|z)\}'\boldsymbol{\beta}$$

which has eliminated $g(z)$. [Robinson \(1988\)](#) proposed this differencing transformation and estimation of $\boldsymbol{\beta}$ from the following OLS regression,

$$y_i - \widehat{E}(y_i|z_i) = \left\{ \mathbf{x}_i - \widehat{E}(\mathbf{x}_i|z_i) \right\}' \boldsymbol{\beta} + v_i \quad (27.4)$$

where the OLS regression does not include a constant and $\widehat{E}(y_i|z_i)$ and $\widehat{E}(x_{1i}|z_i), \dots, \widehat{E}(x_{ki}|z_i)$ are the fitted values from univariate nonparametric

regression such as local constant or local linear. The intercept α is not identified separately from $g(z)$. Nonetheless, it is convenient to center $g(z)$ around an intercept. Then,

$$\hat{g}(z_i) = y_i - \hat{\alpha} - \mathbf{x}'_i \hat{\beta}$$

where $\hat{\alpha}$ is the estimated intercept from OLS regression of y on an intercept, \mathbf{x} and z , and $\hat{\beta}$ is the estimate from (27.4).

The community-contributed `semipar` command ([Verardi and Debarsy 2012](#)) implements this procedure, using the `lpoly` command to obtain the nonparametric fitted values. The default is to use local linear kernel regression with a Gaussian kernel and bandwidth that is calculated by the plugin formula used by the `lpoly` command. The `trim(#)` option trims by discarding those observations where the data on z are sparse, with kernel density estimate below a user-provided threshold.

As an example, for the rest of this chapter, we generate the same data as used in sections 14.4–14.6. The model is $y = 1 + x_1 + x_2 + g(z) + u$, where $g(z) = z + z^2$. Specifically,

```
. * Generated data: y = 1 + 1*x1 + 1*x2 + g(z) + u where g(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()
```

The `semipar` command, with options used to obtain heteroskedastic–robust standard errors and more detailed labeling of the associated figure that is created, yields

```

. * Robinson estimator given unknown g(z) using community-contributed
> command semipar
. semipar y x1 x2, nonpar(z) robust ci title("Partial linear: f(z) against z")
> ytitle("y-b*x and f(z)") xtitle("z")

```

Number of obs = 200
 R-squared = 0.3918
 Adj R-squared = 0.3857
 Root MSE = 1.9925

y	Coefficient	Std. err.	t	P> t	[95% conf. interval]
x1	.9029397	.1516576	5.95	0.000	.6038683 1.202011
x2	.9667308	.1258653	7.68	0.000	.7185221 1.214939

The coefficient estimates are close to the data-generating process (DGP) values of one. The standard errors are asymptotic standard errors that do not control for any estimation error in the first-stage kernel estimation.

For comparison, we consider OLS regression on the DGP model, where we use knowledge that $g(z) = z + z^2$.

```

. * OLS estimation given knowledge of the DGP with g(z) = z + z^2
. regress y x1 x2 z zsq, vce(robust)

```

Linear regression
 Number of obs = 200
 F(4, 195) = 138.56
 Prob > F = 0.0000
 R-squared = 0.6860
 Root MSE = 2.0402

y	Coefficient	Robust		t	P> t	[95% conf. interval]
		std. err.				
x1	.879966	.1539411	5.72	0.000	.5763626	1.183569
x2	.9949839	.1292254	7.70	0.000	.740125	1.249843
z	1.078095	.1345329	8.01	0.000	.8127681	1.343421
zsq	1.065932	.0819577	13.01	0.000	.9042949	1.22757
_cons	.64508	.1844171	3.50	0.001	.2813718	1.008788

The semiparametric model coefficients and their standard errors are very close to those obtained by OLS regression on the DGP model. At the same time, the fit is much better for OLS, with $R^2 = 0.686$ compared with $R^2 = 0.392$, due to the noise in nonparametrically estimating $g(z)$.

The first panel of figure 27.2 presents a plot of $\hat{g}(z)$ on z that is automatically created by the `semipar` command. The relationship appears to

be quadratic, as expected given that the DGP was linear in x_1 and x_2 and $g(z)$, where $g(z)$ was quadratic.

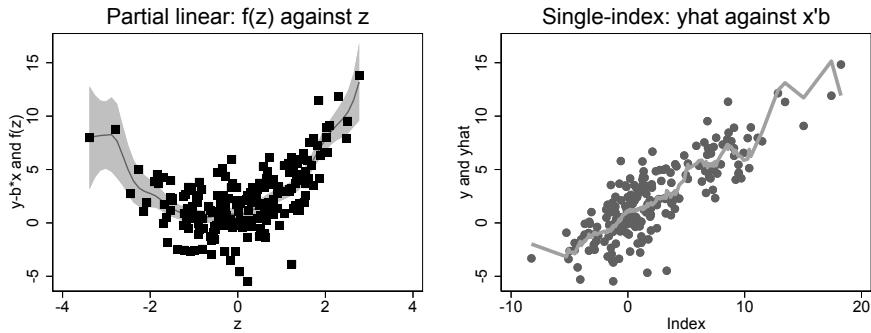


Figure 27.2. Semiparametric regression: Partial linear and single-index models

The `test(#)` option of the `semipar` command computes the test due to [Härdle and Mammen \(1993\)](#) of whether $g(z)$ is a polynomial of specified degree. This test was designed for comparing a kernel regression estimate with a polynomial function and may not be applicable here where $\hat{g}(z)$ is more complicated than simply the fitted values from kernel regression of y on z . From output not given, the `test(1)` option led to $p = 0.009$, the `test(2)` option led to $p = 0.787$, and the `test(3)` option led to $p = 0.860$. The null hypothesis is that $g(z)$ is a polynomial of the specified degree, so at level 0.05, we reject a linear model and do not reject a quadratic or cubic model. We conclude that a quadratic model is appropriate. The test is implemented using a bootstrap. Before each test, we gave command `set seed 10101` and used option `nsims(1000)` to increase the number of bootstraps beyond the default 100.

Note that the results are very dependent on the bandwidths used. In particular, for the Robinson differencing method to yield root- N consistent estimates of β when z is scalar, the first-step kernel regressions should undersmooth, with bandwidths smaller than the usual optimal bandwidths. In end-of-chapter exercise 5, we reproduce the results of the `semipar` command from first principles using the `lpoly` command and find that in this example, the kernel bandwidths were reasonably narrow.

The Robinson differencing method extends to the case where \mathbf{z} is a vector rather than a scalar. The method focuses on estimation of β with $g(z)$ viewed as a nuisance function. Other methods have been developed to fit partial linear models, including using a backfitting algorithm and using penalized splines. An example is given in section [27.8](#) on GAMS.

A series estimator version of the partial linear model can be fit using the `asis()` option of the `npregress series` command; see section [27.3.2](#).

More importantly, methods have been recently developed to use machine learning methods, rather than standard nonparametric methods, when \mathbf{z} is high dimensional; see section [28.8](#).

27.7 Single-index model

We consider the single-index model

$$y = m(\mathbf{x}'\boldsymbol{\beta}) + u$$

where $E(u|\mathbf{x}) = 0$, so the conditional mean of y is $m(\mathbf{x}'\boldsymbol{\beta})$. For example, $E(y|\mathbf{x}) = \exp(\mathbf{x}'\boldsymbol{\beta})$ is a single-index model, as are logit and probit models. A single-index model has the property that if β_j is a times β_k , then the marginal effect on $E(y|\mathbf{x})$ of increasing x_j by one unit is a times the marginal effect of increasing x_k by one unit; see section 13.7.3.

If the function $m(\cdot)$ was known, we could estimate $\boldsymbol{\beta}$ by nonlinear least squares, which minimizes

$$\sum_{i=1}^N \{y_i - m(\mathbf{x}'_i\boldsymbol{\beta})\}^2$$

Now consider the semiparametric case where $m(\cdot)$ is unknown. Then [Ichimura \(1993\)](#) proposed the semiparametric least-squares estimator, which obtains \widehat{m} and $\widehat{\boldsymbol{\beta}}$, which minimize

$$\sum_{i=1}^N w(\mathbf{x}_i) \times \left\{ y_i - \widehat{m}(\mathbf{x}'_i\widehat{\boldsymbol{\beta}}) \right\}^2$$

where $\widehat{m}(\mathbf{x}'_i\widehat{\boldsymbol{\beta}})$ is a kernel regression estimate of $E(y_i|\mathbf{x}'_i\widehat{\boldsymbol{\beta}})$ and $w(\mathbf{x}_i)$ is a trimming parameter equal to one for most observations but equal to zero for those with the largest and smallest values of \mathbf{x}_i .

Not all elements of β are identified. In general, a function $m(a + bx)$ can always be simplified to a function $h(x)$, so the intercept is not identified and b is identified only up to a multiple. It follows that in the current setting, the intercept is not identified, and slope parameters are identified only up to scale. The standard procedure is to set the first slope parameter to one.

The community-contributed `s1s` command ([Barker 2014](#)) implements Ichimura's estimator. We apply this command to the same generated data as used in the preceding partial linear model example, where we use knowledge that the regressors \mathbf{x} are x_1 , x_2 , z , and z_{sq} but do not use knowledge that the DGP specified $m(\mathbf{x}'\beta) = \mathbf{x}'\beta$. We obtain

```

. * Ichimura semiparametric least squares for single-index model
. sls y x1 x2 z zsq, trim(1,99)
initial: SSq(b) = 1632.3579
alternative: SSq(b) = 1418.7811
rescale: SSq(b) = 1113.168
SLS 0: SSq(b) = 1113.168
SLS 1: SSq(b) = 832.81923
SLS 2: SSq(b) = 832.51979
SLS 3: SSq(b) = 832.51701
SLS 4: SSq(b) = 832.51701
    pilot bandwidth
    1.302350774
SLS 0: SSq(b) = 852.36723
SLS 1: SSq(b) = 836.63241
SLS 2: SSq(b) = 813.31622
SLS 3: SSq(b) = 810.40664 (not concave)
SLS 4: SSq(b) = 807.00822
SLS 5: SSq(b) = 802.75174
SLS 6: SSq(b) = 801.87467
SLS 7: SSq(b) = 800.86586
SLS 8: SSq(b) = 800.728
SLS 9: SSq(b) = 800.69764
SLS 10: SSq(b) = 800.69069
SLS 11: SSq(b) = 800.68898
SLS 12: SSq(b) = 800.68864
SLS 13: SSq(b) = 800.68856
SLS 14: SSq(b) = 800.68855

```

Number of obs = 200
root MSE = 2.00086

y	Coefficient	Std. err.	z	P> z	[95% conf. interval]
Index					
x2	1.603177	.3236122	4.95	0.000	.9689089 2.237445
z	1.802992	.327212	5.51	0.000	1.161668 2.444316
zsq	1.526091	.2310128	6.61	0.000	1.073314 1.978867
x1	1	(offset)			

The coefficient of variable x_2 , for example, implies that a 1-unit change in x_2 is associated with a change in $E(y|\mathbf{x})$ that is 1.603 times the change associated with a 1-unit change in x_1 . The need to estimate $m(\cdot)$ leads to considerable loss in efficiency, with z statistics that are 15% to 30% lower than those obtained from OLS regression. For this DGP, with all $\beta_j = 1$, the ratios of all the slope coefficients should equal 1, aside from imprecision in estimation.

The predictions $\hat{m}(\mathbf{x}'\hat{\beta})$ can be obtained using the `ey` option of the `predict` postestimation command, and the `xb` option yields $\mathbf{x}'\hat{\beta}$. The $R^2 = 0.669$ is close to $R^2 = 0.686$ for OLS. The following code computes these quantities and plots $\hat{m}(\mathbf{x}'\hat{\beta})$ and $\mathbf{x}'\hat{\beta}$.

```
. * Create plot of yhat against the index x`b
. predict yhat, ey
. predict Index, xb
. qui correlate y yhat
. display "R-squared = " r(rho)^2
R-squared = .66931948
. twoway (scatter y Index) (line yhat Index, sort lwidth(thick)),
>      title("Single-index: yhat against x`b")
>      xtitle("Index") ytitle("y and yhat") legend(off)
```

The plot given in the second panel of figure [27.2](#) suggests that $m(\cdot)$ is a linear function, as expected given that the DGP had $m(\mathbf{x}'\beta) = \mathbf{x}'\beta$.

27.8 Generalized additive models

The GAM specifies

$$y = g_1(x_1) + \cdots + g_K(x_K) + u$$

where $E(u|\mathbf{x}) = 0$. While the additivity may seem restrictive, the regressors may themselves be interactions. For example, we may have

$$y = g_1(x) + g_2(z) + g_3(x \times z) + u.$$

We consider estimation when the functions $g_1(\cdot), \dots, g_K(\cdot)$ are not specified. Then the GAM has reduced a k -dimensional nonparametric model to K one-dimensional nonparametric models.

The model is fit using a backfitting algorithm. Let $\hat{\alpha} = \bar{y}$, and initially set all $\hat{g}_k(x_{ik}) = 0$. Define the partial residual for the j th regressor to be

$$r_{ij} = y_i - \hat{\alpha} - \sum_{l=1; l \neq j}^K \hat{g}_l(x_{il})$$

For each $j = 1, \dots, K$, perform univariate nonparametric regression of r_{ij} on x_{ij} and update $\hat{g}_j(x_{ij})$ to be the new fitted values from this regression. Repeat this cycle for as long as necessary until the estimates $\hat{g}_j(x_{ij})$ stabilize.

The most commonly used nonparametric method to estimate the components of the GAM are smoothing splines, rather than local constant or local linear kernel-weighted regression. Smoothing splines are presented in section 14.5.3. Briefly, a spline splits the data on the regressor x into segments with the segment boundaries called knots. A cubic spline fits a cubic polynomial between every pair of knots in such a way that the fitted value of y is continuous at each knot and has continuous first and second derivatives at each knot. A cubic smoothing spline lets the knots be the

distinct values of x but then includes a penalty function to avoid overfitting. The extent of the penalty varies with the so-called effective degrees of freedom; higher effective degrees of freedom correspond to a less smooth fit, while a value of one is equivalent to OLS linear regression; see section 14.5.3.

The community-contributed `gam` command ([Royston and Ambler 1998](#)) implements estimation of the GAM using smoothing splines. It provides a front-end to Windows implementation of a slightly modified version of the Fortran program `GAMFIT`, written by [Hastie and Tibshirani \(1990\)](#). It will work only for Windows implementations of Stata.

We apply the `gam` command to the same generated data as used in the preceding Robinson estimator example, where we do not use knowledge that z enters quadratically or that x_1 and x_2 enter linearly. For a reasonably flexible model, we set the effective degrees of freedom for each regressor to 3. We obtain

```
. * Generalized additive model - requires Windows version of Stata
. gam y x1 x2 z, df(3)
200 records merged.

Generalized Additive Model with family gauss, link ident.

Model df     =     10.004
No. of obs =      200
Deviance    =   820.723
Dispersion = 4.31968



| y     | df    | Lin. Coef. | Std. Err. | z      | Gain    | P>Gain |
|-------|-------|------------|-----------|--------|---------|--------|
| x1    | 3.000 | .8474175   | .1804832  | 4.695  | 1.998   | 0.3683 |
| x2    | 3.001 | .9830467   | .1459322  | 6.736  | 2.134   | 0.3444 |
| z     | 3.002 | 1.143934   | .1438065  | 7.955  | 131.768 | 0.0000 |
| _cons | 1     | 2.1644     | .146964   | 14.727 | .       | .      |



Total gain (nonlinearity chisquare) = 135.900 (6.003 df), P = 0.0000


```

The reported effective degrees of freedom are close to, but not exactly identical to, those specified—this is typical and not a cause for concern. The output decomposes the effect of each variable into linear and nonlinear components. For the linear component, the reported coefficient of 0.847 for x_1 , for example, is the slope coefficient from linear OLS regression of the partial residual for x_1 (r_{i1}) on x_1 . All three linear components are statistically significant, indicating that there is a relationship. For the nonlinear component, the `Gain` and `P>Gain` columns provide a test of

whether each $g_j(\cdot)$ is nonlinear. At level 0.05, we reject nonlinearity in x_1 and x_2 , while there is clearly nonlinearity in z . These results are all expected given the DGP.

The model fit is good. The statistic dispersion is $\sum_{i=1}^N (y_i - \hat{y}_i)^2 / (N - df)$, where here $df = 10.004$, so $\sum_{i=1}^N (y_i - \hat{y}_i)^2 = 4.31968 \times (200 - 10.004) = 820.72$. And $\sum_{i=1}^N (y_i - \bar{y})^2 = 2584.86$. So $R^2 = 0.682$, very close to 0.686 from OLS regression of y on x_1 , x_2 , z , and $z\text{sq}$.

The `gam` command stores the fitted values \hat{y}_i in variable `GAM_mu`. Let variable x_1 be the first regressor, denoted x_1 . Then the variable `s_x1` stores $\hat{g}_1(x_{i1})$, the variable `e_x1` stores the standard error of $\hat{g}_1(x_{i1})$, and the variable `r_x1` stores the partial residual r_{i1} . So a plot of `s_x1` on x_1 is a plot of $\hat{g}_1(x_{i1})$ on x_{i1} .

The `gamplot` postestimation command plots for a specified regressor, say, x_j , and the fitted smooth $\hat{g}_j(x_{ij})$ against x_{ij} , along with a 95% confidence interval and a scatterplot of the partial residual r_{ij} against x_{ij} . Creating these plots separately for each of the three regressors, we have

```
. * Plot fitted smooth (and partial residual) against x for each regressor x
. qui gam y x1 x2 z, df(3)
. gamplot x1, saving(graph1.gph, replace)
(file graph1.gph saved)
. gamplot x2, saving(graph2.gph, replace)
(file graph2.gph saved)
. gamplot z, saving(graph3.gph, replace)
(file graph3.gph saved)
. graph combine graph1.gph graph2.gph graph3.gph,
>      iscale(1.2) rows(1) ysize(2) xsize(6)
```

For the first panel in figure 27.3, the solid line is a plot of the fitted smooth `s_x1` on x_1 ; the upper confidence band is a plot of `s_x1 + 1.96 × e_x1` on x_1 ; and the scatterplot is of the partial residual `r_x1` on x_1 . The panels clearly show a linear relationship for x_1 and x_2 and a quadratic relationship for z .

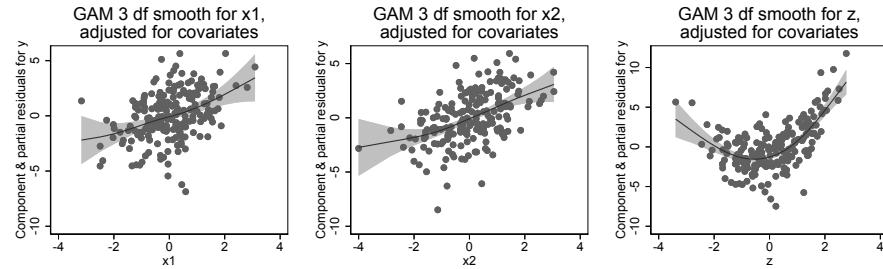


Figure 27.3. GAM: Partial residuals and fitted $g(\cdot)$ against regressors x_1 , x_2 , and z

The `gam` command provides an alternative method for fitting the partial linear model by setting the effective degrees of freedom to one for the variables that enter linearly and using a cubic smoothing spline with effective degrees of freedom three, for example, for the variables that enter nonlinearly. For the current example, we have

```
. * Partial linear model y = b0+b1*x1+b2*x2+g(z) estimated using GAM
. gam y x1 x2 z, df(x1:1, x2:1, z:3)
200 records merged.

Generalized Additive Model with family gauss, link ident.

Model df      =     6.002
No. of obs   =      200
Deviance     =    836.24
Dispersion   =    4.31057



| y     | df    | Lin. Coef. | Std. Err. | z      | Gain    | P>Gain |
|-------|-------|------------|-----------|--------|---------|--------|
| x1    | 1     | .8482625   | .1802929  | 4.705  | .       | .      |
| x2    | 1     | .9720385   | .1457786  | 6.668  | .       | .      |
| z     | 3.002 | 1.137985   | .1436547  | 7.922  | 137.572 | 0.0000 |
| _cons | 1     | 2.1644     | .146809   | 14.743 | .       | .      |


```

Total gain (nonlinearity chisquare) = 137.572 (2.002 df), P = 0.0000
. display "R2 = " 1 - e(disp)*e(tdf) / 2584.6
R2 = .67645209

The R^2 is 0.676 compared with 0.392 using Robinson's differencing estimator and 0.686 using OLS.

27.9 Additional resources

The standard econometrics references for nonparametric and semiparametric regression are [Pagan and Ullah \(1999\)](#) and [Li and Racine \(2007\)](#). [Cameron and Trivedi \(2005, chap. 9\)](#) and [Hansen \(2022, chaps. 19 and 20\)](#) provide a briefer treatment.

The `npregress kernel` command computes kernel-weighted local constant and local linear regression with single or multiple regressors and is a much richer command than `lpoly`. The `npregress series` command implements nonparametric polynomial and spline regression. The most commonly used semiparametric models can be implemented using the following community-contributed commands detailed in the main text: `semipar` for the partial linear model, `s1s` for the single-index model, and `gam` for the GAM. The last command works only on Windows computers. See also chapter [28](#), which presents machine learning methods for prediction such as lasso, neural networks, regression trees, and random forests, and for estimation of the partial linear model.

27.10 Exercises

1. Consider data on (y, x) with sample values $(3, 2)$, $(2, 4)$, $(6, 6)$, $(10, 8)$, and $(8, 10)$. Using the `npregress` command, obtain the local constant estimator of y given x using the `epan2` kernel and option `meanbwidht(3, copy)`, which sets the bandwidth $h = 3$. Use the `predict` command to obtain the fitted values. Now, manually compute the local constant estimate of y when $x = 6$, and verify that you obtain the same estimate as that obtained from the `npregress` command.
2. Use the generated dataset of section [27.6](#), and perform nonparametric regression of y on z . First, perform local linear regression using the `npregress` command with `epan2` kernel. Repeat using the `lpoly` command with the `at(z)` and `degree(1)` options. Compare the fitted values and explain any difference. Now, again use the `npregress` command, but set the bandwidth to be the same as that used by the `lpoly` command. Compare the fitted values and explain any difference. Now, repeat the entire exercise but using local constant regression, rather than local linear regression. You will need to use the `noderivative` option of the `npregress` command.
3. Use `mus204mabeldecomp.dta`, which has Australian data on log of doctor annual earnings (`logyearn`) and on annual hours of work (`yhrs`). Form variable `logyhrs` ($= \ln(yhrs)$), and to speed computation, obtain a 2% subsample using commands `set seed 10101` and `keep if runiform() < 0.02`. For visual analysis, provide a graph with a scatterplot of `logyearn` on `logyhrs`, a fitted linear regression, and a fitted local linear regression using the `lpoly` command. Does the relationship appear to be linear? Next, obtain local linear estimates using command `npregress kernel logyearn logyhrs kernel(epan2) vce(bootstrap, seed(10101) reps(100))`. Is the estimated average effect consistent with an hours elasticity of earnings equal to one? Compare the estimated average effect to that obtained from OLS regression. Finally, following the local linear regression, obtain predicted means of `logyearn` when variable `logyhrs` is evaluated at values corresponding to annual hours of 1,000, 2,000, and 3,000.

4. Continue with the data of the previous question. We will add as controls experience (`expr`), experience-squared (`exprsq`), gender (`female`), and presence in household of child of age under five (`childu5`). Perform local linear regression of `logyearn` on `logyhrs` and these additional controls—note that the discrete regressor needs to be included as `i.female`. Is the estimated average effect consistent with an hours elasticity of earnings equal to one? Does gender make a big difference? Compare the estimated average effects of regressors with those obtained from OLS regression. Finally, following the local linear regression obtain predicted means of `logyearn` when variable `logyhrs` is evaluated at values corresponding to annual hours of 1,000, 2,000, and 3,000.
5. Implement the `semipar` example in section [27.6](#) manually as follows. Use command `lpoly` with options `at(z)`, `degree(1)`, and `kernel(gaussian)` to perform local linear regression of `x1` on `z`. Compute a variable `u_x1` that is the residual from this regression. Repeat for regression of `x2` on `z` and `y` on `z` to yield residuals `u_x2` and `u_y`. OLS regress `u_y` on `u_x1` and `u_x2`, without an intercept, and obtain heteroskedastic–robust standard errors. Compare your answer with that from command `semipar y x1 x2, nonpar(z) nograph robust`. Next, we want to reproduce the graph produced by the `semipar` command. Following the preceding OLS regression, generate the variable `ylessxb = y - b[u_x1]*x1 - b[u_x2]*x2`. Produce a scatterplot of `ylessxb` on `z`, along with the fitted values from local linear regression of `ylessxb` on `z` using command `lpoly` with options `at(z)`, `degree(1)`, and `kernel(gaussian)`.

Chapter 28

Machine learning for prediction and inference

28.1 Introduction

Microeconometrics studies tend to focus on estimation of one or more regression model parameters β and subsequent statistical inference on β or relevant marginal effects that are a function of β .

A quite different purpose of statistical analysis is prediction. For example, we may wish to predict the probability of 12-month survival following hip or knee replacement surgery. In that case, we are interested in obtaining a good prediction \hat{y} of the dependent variable y .

In principle, nonparametric methods such as kernel regression can provide a flexible way to obtain predictions. But these methods suffer from the curse of dimensionality if there are many potential regressors. The machine learning literature has proposed a wide range of alternative techniques for prediction, where the term “machine learning” is used because the machine, here the computer, selects the best predictor using only the data at hand, rather than via a model specified by the researcher, who has detailed knowledge of the specific application.

Machine learning entails data mining that can lead to overfitting the sample at hand. To guard against overfitting, one assesses models on the basis of out-of-sample prediction using cross-validation (cv) or by penalizing model complexity using information criteria or other penalty measures. We begin by presenting cv and penalty measures.

We then present various techniques for prediction that are used in the machine learning literature. We focus on shrinkage estimators, notably the lasso. Lasso, originally an acronym for “least absolute shrinkage and selection operator”, is now considered a word. Additionally, a brief review is provided of principal components for dimension reduction, as well as neural networks, regression trees, and random forests for flexible nonlinear models. There is no universal best method for prediction, though for some specific data types, such as images or text, one method may work particularly well. Often, an ensemble prediction that is a weighted average of predictions from different methods performs best.

The machine learning literature has focused on prediction of y . The recent econometrics literature has developed methods that use machine learning methods as an intermediate input to the ultimate goal of estimating and performing inference on model parameters of interest. This is a very active area of research.

A leading inference example of machine learning methods is inference on α in the partial linear model $y = \mathbf{d}'\alpha + g(\mathbf{x}) + u$, where machine learning methods are used to determine the best function of controls \mathbf{x} . A second leading example is instrumental-variables (IV) estimation with many potential instruments or controls, or both. Then we wish to select only a few variables to avoid the weak instrument problems that can arise with many instruments. These and related examples cover many applied microeconomics applications, and the development of methods for valid inference with machine learning as an input promises to be revolutionary.

The econometrics inference literature to date has emphasized use of the lasso, and Stata 16 has introduced a suite of commands for prediction and inference using the lasso. For these reasons, we emphasize the lasso. We expect that additional methods, notably, random forests and neural nets, will be increasingly used in microeconomic studies.

28.2 Measuring the predictive ability of a model

There are several ways to measure the predictive ability of a model. Ideally, such measures penalize model complexity and control for in-sample overfitting.

Traditionally, econometricians have used penalty measures such as in-sample adjusted R^2 and in-sample information criteria. The machine learning literature instead emphasizes out-of-sample predictive ability using cv . An introductory treatment is given in [James et al. \(2021\)](#), chaps. 5, 6.1).

28.2.1 Generated data example

The example used in much of this chapter is one where the continuous dependent variable y is regressed on three correlated normally distributed regressors, denoted x_1 , x_2 , and x_3 . The actual data-generating process (DGP) for y is a linear model with an intercept and x_1 alone. Many of the methods can be adapted for other types of data such as binary outcomes and counts.

The data are generated using commands presented in chapter 5, notably, the command `drawnorm` to generate correlated normally distributed regressors with correlation 0.5 and the `rnormal()` function to obtain normally distributed errors. The DGP for y is a linear model with intercept 2 and slope coefficient 1 for variable x_1 . We have

```
. * Generate three correlated variables (rho = 0.5) and y linear only in x1
. qui set obs 40
. set seed 12345
. matrix MU = (0,0,0)
. scalar rho = 0.5
. matrix SIGMA = (1,rho,rho \ rho,1,rho \ rho,rho,1)
. drawnorm x1 x2 x3, means(MU) cov(SIGMA)
. generate y = 2 + 1*x1 + rnormal(0,3)
```

Summary statistics and correlations for the variables are

```

. * Summarize data
. summarize

  Variable |   Obs    Mean   Std. dev.    Min    Max
  _____|_____|_____|_____|_____|_____|_____
    x1     |   40    .3337951    .8986718   -1.099225   2.754746
    x2     |   40    .1257017    .9422221   -2.081086   2.770161
    x3     |   40    .0712341    1.034616   -1.676141   2.931045
    y      |   40    3.107987    3.400129   -3.542646  10.60979

. correlate
(obs=40)

  x1    x2    x3    y
  _____|_____|_____|_____|_____
  x1  1.0000
  x2  0.5077  1.0000
  x3  0.4281  0.2786  1.0000
  y   0.4740  0.3370  0.2046  1.0000

```

Ordinary least-squares (OLS) estimation of y on x_1 , x_2 , and x_3 yields the following:

```

. * OLS regression of y on x1-x3
. regress y x1 x2 x3, vce(robust)

Linear regression                               Number of obs =        40
                                                F(3, 36)      =       4.91
                                                Prob > F    =     0.0058
                                                R-squared    =     0.2373
                                                Root MSE     =     3.0907

```

y	Coefficient	Robust	t	P> t	[95% conf. interval]
		std. err.			
x1	1.555582	.5006152	3.11	0.004	.5402873 2.570877
x2	.4707111	.5251826	0.90	0.376	-.5944086 1.535831
x3	-.0256025	.6009393	-0.04	0.966	-1.244364 1.193159
_cons	2.531396	.5377607	4.71	0.000	1.440766 3.622025

Only variable x_1 is statistically significant at level 0.05. This is very likely given the DGP depends only on x_1 , but it is by no means certain. Because of randomness, we expect that variable x_2 or variable x_3 will be statistically significant at level 0.05 in 5% of similarly generated datasets.

28.2.2 Mean squared error

In general, we predict at point \mathbf{x}_0 using $\hat{y}_0 = \hat{g}(\mathbf{x}_0)$, where for OLS $\hat{y}_0 = \mathbf{x}'_0 \hat{\beta}$. We wish to estimate the expected prediction error $E\{(y_0 - \hat{y}_0)^2\}$.

The standard criterion used for a continuous dependent variable is minimization of the mean squared error (MSE) of the predictor

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

If the MSE is computed in sample, it underestimates the true prediction error. One way to see this underestimation is that with N independent regressors, including the intercept, OLS necessarily produces a perfect fit with $R^2 = 1$ and $\text{MSE} = 0$. By contrast, the true prediction error will generally be greater than 0.

A second way to see this underestimation is to note that if $\mathbf{y} = \mathbf{X}\beta + \mathbf{u}$, then the OLS residual vector $\hat{\mathbf{u}} = \mathbf{y} - \mathbf{X}\hat{\beta} = (\mathbf{I} - \mathbf{M})\mathbf{u}$, where $\mathbf{M} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}$. Because $(\mathbf{I} - \mathbf{M}) < \mathbf{I}$ in the matrix sense, it follows that on average $|\hat{u}_i| < |u_i|$, so the OLS residual on average is smaller than the true unknown error. For similar reasons, with independent homoskedastic errors, the unbiased estimator of σ^2 is $s^2 = \{1/(N - K)\} \sum_{i=1}^N (y_i - \hat{y}_i)^2$ and not the smaller $(1/N) \sum_{i=1}^N (y_i - \hat{y}_i)^2$.

Several methods seek to adjust MSE for model size. These include information criteria that penalize MSE for model size and CV measures that fit the model on a subsample and compute MSE for the remainder of the sample.

We focus on using MSE as the measure of predictive ability. For continuous data, other measures can be used such as the mean absolute error $(1/N) \sum_{i=1}^N |y_i - \hat{y}_i|$. For likelihood-based models, the log likelihood may be used. For generalized linear models, the deviance may be used. For binary outcomes, the number of incorrect classifications is commonly used if interest lies in predicting the actual outcome y , rather than $\Pr(y = 1)$.

28.2.3 Information criteria and related penalty measures

Two standard measures that penalize model fit for model size are Akaike's information criterion (AIC) and the Bayesian information criterion (BIC). The general formulas for these measures were presented in section 13.8.2.

Specializing to the classical linear regression model under independent and identically distributed normal errors, the fitted log likelihood equals $N \ln 2\pi + N + \ln \text{MSE}$, leading to

$$\begin{aligned} \text{AIC} &= N \ln 2\pi + N + \ln \text{MSE} + 2K \\ \text{BIC} &= N \ln 2\pi + N + \ln \text{MSE} + (\ln N) \times K \end{aligned}$$

where K is the number of regressors, including the intercept. Models with smaller AIC and BIC are preferred, so AIC and BIC are penalized measures of MSE. BIC has a larger penalty for model size than AIC, so BIC leads to smaller models and might be preferred when more parsimonious models are desired.

A related information measure is Mallow's C_p measure,

$$C_p = (N \times \text{MSE}/\hat{\sigma}^2) - N + 2K$$

where $\hat{\sigma}^2 = \sum_{i=1}^N (y_i - \tilde{y}_i)^2 / (N - p)$ and \tilde{y}_i is the OLS prediction from the largest model under consideration that has p regressors, including the intercept. Models with smaller C_p are preferred. Selecting a model on the basis of minimum C_p is asymptotically equivalent to minimizing AIC.

Another penalty measure that is often used is adjusted R^2 , denoted \bar{R}^2 , which can be expressed as

$$\bar{R}^2 = 1 - \frac{\text{MSE} \times N/(N - K)}{\text{TSS}/(N - 1)}$$

where $\text{TSS} = \sum_{i=1}^N (y_i - \bar{y}_i)^2$. Again, MSE is penalized for model size because \bar{R}^2 is a decreasing function of K . However, this penalty is relatively

small. In the classical linear regression model with homoskedastic normally distributed errors, it can be shown that, for nested models, choosing the larger model if it has higher \bar{R}^2 is equivalent to choosing the larger model if the F statistic for testing joint significance of the additional regressors exceeds one. The usual critical values used in such a test are substantially greater than one.

To enable comparison of all eight possible models that are linear in parameters and regressors, we first define the regressor lists for each model.

```
. * Regressor lists for all possible models
. global xlist1
. global xlist2 x1
. global xlist3 x2
. global xlist4 x3
. global xlist5 x1 x2
. global xlist6 x2 x3
. global xlist7 x1 x3
. global xlist8 x1 x2 x3
```

The following code provides a loop that estimates each model and computes the various penalized fit measures. Note that the global macro defining the k th regressor list needs to be referenced as `${xlist}'k'` rather than simply `${xlist}'k'`. We have

```

. * Full-sample estimates with AIC, BIC, Cp, R2adj penalties
. qui regress y $xlist8
. scalar s2full = e(rmse)^2 // Needed for Mallows Cp
. forvalues k = 1/8 {
2.     qui regress y ${xlist`k`}
3.     scalar mse`k` = e(rss)/e(N)
4.     scalar r2adj`k` = e(r2_a)
5.     scalar aic`k` = -2*e(ll) + 2*e(rank)
6.     scalar bic`k` = -2*e(ll) + e(rank)*ln(e(N))
7.     scalar cp`k` = e(rss)/s2full - e(N) + 2*e(rank)
8.     display "Model " "${xlist`k`}" _col(15) " MSE=" %6.3f mse`k`
>         " R2adj=" %6.3f r2adj`k` " AIC=" %7.2f aic`k`
>         " BIC=" %7.2f bic`k` " Cp=" %6.3f cp`k`
9. }
Model      MSE=11.272 R2adj= 0.000  AIC= 212.41 BIC= 214.10 Cp= 9.199
Model x1    MSE= 8.739 R2adj= 0.204  AIC= 204.23 BIC= 207.60 Cp= 0.593
Model x2    MSE= 9.992 R2adj= 0.090  AIC= 209.58 BIC= 212.96 Cp= 5.838
Model x3    MSE=10.800 R2adj= 0.017  AIC= 212.70 BIC= 216.08 Cp= 9.224
Model x1 x2  MSE= 8.598 R2adj= 0.196  AIC= 205.58 BIC= 210.64 Cp= 2.002
Model x2 x3  MSE= 9.842 R2adj= 0.080  AIC= 210.98 BIC= 216.05 Cp= 7.211
Model x1 x3  MSE= 8.739 R2adj= 0.183  AIC= 206.23 BIC= 211.29 Cp= 2.592
Model x1 x2 x3 MSE= 8.597 R2adj= 0.174  AIC= 207.57 BIC= 214.33 Cp= 4.000

```

The MSE, which does not penalize for model size, is smallest for the largest model with all three regressors. The penalized measures all favor the model with an intercept and x_1 as the only regressors. More generally, different penalties may favor different models.

28.2.4 The `splitsample` command

The preceding example used the same sample for both estimation and measurement of model fit. `cv` instead fits the model on one sample, called a training sample, and measures predictive ability based on a different sample, called a test sample or holdout sample or validation sample. This approach can be applied to a range of models and to loss functions other than `MSE`.

Mutually exclusive samples of prespecified size can be generated using the command `splitsample`, which creates a variable identifying the different samples. For example, the sample can be split into five equally sized mutually exclusive samples as follows:

```
. * Split sample into five equal-size parts using splitsample command
. splitsample, nsplit(5) generate(snum) rseed(10101)
. tabulate snum
```

snum	Freq.	Percent	Cum.
1	8	20.00	20.00
2	8	20.00	40.00
3	8	20.00	60.00
4	8	20.00	80.00
5	8	20.00	100.00
Total	40	100.00	

For replicability, the `rseed()` option is used. The variable `snum` identifies the five samples. The alternative `split()` option allows splitting in specified ratios. For example, `split(1 1 2)` will split the sample into subsamples of, respectively, 25%, 25%, and 50% of the sample. The `cluster()` option enables sample splitting by cluster. If data are missing, then it is best to include as an argument a list of relevant variables to ensure that the splits are on the sample with nonmissing observations. This is especially important if, for example, observations with missing values appeared at the end of the dataset.

28.2.5 Single-split cross-validation

We begin with the simplest approach of single-split validation. We randomly divide the original sample into two parts: a training sample on which the model will be fit and a test sample that will be used to assess the fit of the model.

It is common to use a larger part of the original sample for estimation and a smaller part for assessing predictive ability. The following code creates an indicator variable for a training sample of 80% of the data (`dtrain==1`) and a test sample of 20% of the data (`dtrain==0`).

```

. * Form indicator for training data (80% of sample) and test data (20%)
. splitsample, split(1 4) values(0 1) generate(dtrain) rseed(10101)
. tabulate dtrain

```

dtrain	Freq.	Percent	Cum.
0	8	20.00	20.00
1	32	80.00	100.00
Total	40	100.00	

We fit each of the 8 potential regression models on the 32 observations in the training sample and compute the MSE separately for the 32 observations in the training sample (an in-sample MSE) and for the 8 observations in the test sample (an out-of-sample MSE).

```

. * Single-split validation - training and test MSE for the 8 possible models
. forvalues k = 1/8 {
    2.     qui reg y ${xlist`k`} if dtrain==1
    3.     qui predict y`k`hat
    4.     qui gen y`k`errorsq = (y`k`hat - y)^2
    5.     qui sum y`k`errorsq if dtrain == 1
    6.     scalar mse`k`train = r(mean)
    7.     qui sum y`k`errorsq if dtrain == 0
    8.     qui scalar mse`k`test = r(mean)
    9.     display "Model " "${xlist`k`}" _col(16)
    >         " Training MSE = " %7.3f mse`k`train " Test MSE = " %7.3f mse`k`test
  10. }
Model          Training MSE =  10.124 Test MSE =  16.280
Model x1        Training MSE =   7.478 Test MSE = 13.871
Model x2        Training MSE =   8.840 Test MSE = 14.803
Model x3        Training MSE =   9.658 Test MSE = 15.565
Model x1 x2      Training MSE =   7.288 Test MSE = 13.973
Model x2 x3      Training MSE =   8.668 Test MSE = 14.674
Model x1 x3      Training MSE =   7.474 Test MSE = 13.892
Model x1 x2 x3   Training MSE =   7.288 Test MSE = 13.980
. drop y*hat y*errorsq

```

As expected, the in-sample MSE (where we normalize by N) decreases as regressors are added and is minimized at 7.288 when all 3 regressors are included.

But when we instead consider the out-of-sample MSE, we find that this is minimized at 13.871 when only x_1 is a regressor. Indeed, the model with all three regressors has the fifth-highest out-of-sample MSE, due to in-sample overfitting.

28.2.6 K-fold cross-validation

The results from single-split validation depend on how the sample is split. For example, in the current example, different sample splits due to different seeds can lead to out-of-sample MSE being minimized by models other than that with x_1 alone as regressor. K -fold cv reduces this limitation by forming more than one split of the full sample.

Specifically, the sample is randomly divided into K groups or folds of approximately equal size. In turn, one of the K folds is used as the test dataset, while the remaining $K - 1$ folds are used as the training set. Thus, when fold 1 is the test dataset, the model is fit on folds 2 to K ; when fold 2 is the test dataset, the model is fit on fold 1 and folds 3 to K ; and so on. The following shows the case $K = 5$.

	Fit on folds	Test on fold
$j = 1$	2,3,4,5	1
$j = 2$	1,3,4,5	2
$j = 3$	1,2,4,5	3
$j = 4$	1,2,3,5	4
$j = 5$	1,2,3,4	5

Then the cv measure is the average of the K MSEs,

$$\text{CV}_K = \frac{1}{K} \sum_{j=1}^K \text{MSE}_{(j)}$$

where $\text{MSE}_{(j)}$ is the MSE for fold j based on OLS estimates obtained by regression using all data except fold j .

As the number of folds increases, the training set size increases, so bias decreases. At the same time, the fitted models overlap, so test set predictions

are more highly correlated, leading to greater variance in the estimate of the expected prediction error $E\{(y_0 - \hat{y}_0)^2\}$. The consensus is that $K = 5$ or $K = 10$ provides a good balance between bias and variance; most common is to set $K = 10$.

With so few observations, we use $K = 5$ and obtain the MSE for each fold.

```
. * Five-fold CV example for model with all regressors
. splitsample, nsplit(5) generate(foldnum) rseed(10101)
. matrix allmses = J(5,1,.)
. forvalues i = 1/5 {
2.     qui reg y x1 x2 x3 if foldnum != `i'
3.     qui predict y`i'hat
4.     qui gen y`i'errorsq = (y`i'hat - y)^2
5.     qui sum y`i'errorsq if foldnum ==`i'
6.     matrix allmses[`i',1] = r(mean)
7. }
. matrix list allmses
allmses[5,1]
      c1
r1  13.980321
r2  6.4997357
r3  9.3623792
r4  6.413401
r5  12.23958
```

To obtain the CV_5 measure, we convert the matrix `allmses` to a variable and obtain its mean.

```
. * Compute the average MSE over the five folds and standard deviation
. svmat allmses, names(vallmses)
. qui sum vallmses1
. display "CV5 = " %5.3f r(mean) " with st. dev. = " %5.3f r(sd)
CV5 = 9.699 with st. dev. = 3.389
```

The resulting CV_5 measure is 9.699. The MSE's do vary considerably over the five folds with standard deviation 3.389, which is large relative to the average.

The community-contributed `crossfold` command ([Daniels 2012](#)) performs K -fold cv. Applying this to all 8 potential models, using $K = 5$

and the same split for each model, we obtain

```
. * Five-fold CV measure for all possible models
. forvalues k = 1/8 {
2.     set seed 10101
3.     qui crossfold regress y ${xlist`k`}, k(5)
4.     matrix RMSEs`k` = r(est)
5.     svmat RMSEs`k`, names(rmse`k`)
6.     qui generate mse`k` = rmse`k``^2
7.     qui sum mse`k`
8.     scalar cv`k` = r(mean)
9.     scalar sdcv`k` = r(sd)
10.    display "Model " "${xlist`k`}" _col(16) " CV5 = " %7.3f cv`k`
>      " with st. dev. = " %7.3f sdcv`k`
11. }
Model          CV5 = 11.960 with st. dev. = 3.561
Model x1        CV5 = 9.138 with st. dev. = 3.069
Model x2        CV5 = 10.407 with st. dev. = 4.139
Model x3        CV5 = 11.776 with st. dev. = 3.272
Model x1 x2     CV5 = 9.173 with st. dev. = 3.367
Model x2 x3     CV5 = 10.872 with st. dev. = 4.221
Model x1 x3     CV5 = 9.639 with st. dev. = 2.985
Model x1 x2 x3  CV5 = 9.699 with st. dev. = 3.389
```

The `crossfold` command reports for each fold root mean squared error (RMSE) the square root of the MSE, rather than the MSE. To compute the CV_5 measure, we retrieve the RMSEs stored in matrix `r(est)` and calculate the average of the squares of the RMSEs.

The CV measure is lowest for the model with `x1` the only regressor. However, it is only slightly higher in the model with both `x1` and `x2` included. Recall that the folds are randomly chosen, so that with different seed, we would obtain different folds, different CV values, and hence might find that, for example, the model with both `x1` and `x2` included has the minimum CV.

Because of the randomness in K -fold CV, some studies use a one standard-error rule that chooses the smallest model with CV within one standard deviation of the model with minimum CV. Applying that rule in this particular example favors (marginally) an intercept-only model because $9.138 + 3.069 = 12.207 > 11.960$.

Once the preferred model is obtained by cv, it is fit using the entire sample or, potentially, a new sample. Note that the data mining to obtain a preferred model introduces issues similar to those raised by pretest bias; see section 11.3.8. Section [28.8](#) provides examples of special settings, methods, and assumptions for which it is possible to ignore the data mining.

cv is easily adapted to estimators other than OLS and to other loss functions such as mean absolute error $(1/N)\sum_{i=1}^N |y_i - \hat{y}_i|$. Information criteria have the advantage of being less computationally demanding and can yield results not too dissimilar from those obtained using cv.

28.2.7 Leave-one-out cross-validation

Leave-one-out cross-validation (LOOCV) is the special case of K -fold cv with $K = N$. Then N models are fit; in each model $(N - 1)$ observations are used in training, and the remaining observation is used for validation. So we drop each observation in turn, fitting a model without that observation and then using the fitted model to predict the dropped observation.

The community-contributed `loocv` command ([Barron 2014](#)) implements LOOCV. Note, however, that it is quite slow because it is written to apply to any Stata estimation command and does not take advantage of the great computational savings that are possible in the special case of OLS.

For the model with `x1` the only regressor, we obtain

```
. * LOOCV
. loocv regress y x1
```

Leave-One-Out Cross-Validation Results

Method	Value
Root Mean Squared Errors	3.0989007
Mean Absolute Errors	2.5242994
Pseudo-R2	.15585569

```
. display "LOOCV MSE = " r(rmse)^2
LOOCV MSE = 9.6031853
```

The MSE from LOOCV is $3.0989^2 = 9.603$, compared with the preceding CV5 measure of 9.138 in the model with just x_1 as regressor.

LOOCV is not as good for measuring global fit because the N folds are highly correlated with each other, leading to higher variance than if $K = 5$ or $K = 10$. LOOCV is used especially for nonparametric regression where concern is with local fit; see section [27.2.4](#). Assuming a correctly specified likelihood model, model selection on the basis of LOOCV is asymptotically equivalent to using AIC.

28.2.8 Best subsets selection and stepwise selection

The best subsets method sequentially determines the best-fitting model for models with one regressor, with two regressors, and so on up to all p potential regressors. Then K -fold cross-validated MSE or a penalty measure such as AIC or BIC is computed for these p best-fitting models of different sizes. In theory, there are 2^p models to fit, but this is greatly reduced by using a method called the leaps-and-bounds algorithm.

Stepwise selection methods, introduced in section 11.3.7, entail less computation than the best subsets method. For example, with p potential regressors, the stepwise forward procedure requires $p + (p - 1) + \dots + 1 = p(p + 1)/2$ regressions.

The community-contributed `vselect` command ([Lindsey and Sheather 2010](#)) implements best subsets and stepwise selection methods for OLS regression with predictive ability measured using any of adjusted R^2 , AIC, BIC, or AICC, where AICC is a bias-corrected version of AIC that equals $AIC + 2(K + 1)(K + 2)/(N - K - 2)$. The `vselect` command, however, does not cover K -fold cross-validated MSE.

The default for the `vselect` command is to use the best subsets method. We obtain

```

. * Best subset selection with community-contributed command vselect
. vselect y x1 x2 x3, best

Response :          y
Selected predictors: x1 x2 x3

Optimal models:

# Preds      R2ADJ      C      AIC      AICC      BIC
  1  .2043123  .5925225  204.2265  204.8932  207.6042
  2  .1959877  2.002325  205.5761  206.7189  210.6427
  3  .1737073           4  207.5735  209.3382  214.329

predictors for each model:
1  :  x1
2  :  x1 x2
3  :  x1 x2 x3

```

For models of a given size, all measures reduce to minimizing MSE, while the various models give different penalties for increased model size. The best-fitting models with one, two, and three regressors are those with, respectively, regressors x_1 , (x_1, x_2) , and (x_1, x_2, x_3) . All the penalized measures favor the model with just x_1 and an intercept as regressor.

The `forward` (or `backward`) option of the `vselect` command implements forward (or backward) selection. Then one additionally needs to specify which of the various penalty measures is used as model-selection criterion. The `fix()` option of the `vselect` command enables specifying regressors that should be included in all models, and the command permits weighted regression.

The community-contributed `gvselect` command ([Lindsey and Sheather 2015](#)) implements best subsets selection for any Stata command that reports a fitted log likelihood. Then the best-fitting model of a given size is that with the highest-fitted log likelihood, and the best model overall is that with the smallest AIC or BIC.

28.3 Shrinkage estimators

The linear model can be made quite flexible by including as regressors transformations of underlying variables such as polynomials and interactions. Nonetheless, the machine learning literature has introduced other models and other estimation methods that can predict better than OLS.

In this section, we present shrinkage estimators, most notably, the lasso, that shrink parameter estimates toward zero. The resultant reduction in variability may be sufficiently large enough to offset the induced bias leading to lower MSE.

To see this potential gain, consider a scalar unbiased estimator $\hat{\theta}$ with $E(\hat{\theta}) = \theta$ and $\text{Var}(\hat{\theta}) = v$. Then $\text{MSE}(\hat{\theta}) = v$ because the MSE equals variance plus squared bias

$$E \left\{ (\hat{\theta} - \theta)^2 \right\} = E \left[\left\{ \hat{\theta} - E(\hat{\theta}) \right\}^2 \right] + \left\{ E(\hat{\theta}) - \theta \right\}^2$$

Now, define the shrinkage estimator $\tilde{\theta} = a\hat{\theta}$, where $0 \leq a \leq 1$. Then $\text{Var}(\tilde{\theta}) = \text{Var}(a\hat{\theta}) = a^2v$ and $\text{Bias}(\tilde{\theta}) = E(\tilde{\theta}) - \theta = (a - 1)\theta$, so $\text{MSE}(\tilde{\theta}) = a^2v + (a - 1)^2\theta^2$. In the case that $\tilde{\theta}$ shrinks all the way to 0 ($a = 0$), $\tilde{\theta}$ has lower MSE than $\hat{\theta}$ if $\theta^2 < v$. And if $\tilde{\theta} = 0.9\hat{\theta}$, then $\tilde{\theta}$ has lower MSE than $\hat{\theta}$ for $\theta^2 < 19v$. The potential reduction in MSE of $\tilde{\theta}$ carries over directly to the predictor $\tilde{y} = x_0\tilde{\theta}$.

Shrinkage methods are also called penalized or regularized methods. Many shrinkage estimators can also be interpreted in a Bayesian framework as weighted sums of a specified prior and sample maximum-likelihood estimator. In some other cases, the shrinkage estimator may be a limiting form of such an estimator.

We focus on shrinkage for linear regression with MSE loss. But shrinkage estimators can be applied to other settings with different loss

functions, such as maximum likelihood estimation for the logit model.

We present the leading shrinkage estimators: ridge regression shrinks all parameters toward zero, the lasso sets some parameters to zero, while other parameters are shrunk toward zero, and the elastic net combines ridge regression and lasso. An introductory treatment is given in [James et al. \(2021\)](#), chap. 6.2).

28.3.1 Ridge regression

The ridge estimator of [Hoerl and Kennard \(1970\)](#), also known as Tikhonov regularization, is a biased estimator that reduces MSE by retaining all regressors but shrinking parameter estimates toward zero.

The ridge estimator $\hat{\beta}_\lambda$ of β minimizes

$$Q_\lambda(\beta) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{x}'_i \beta)^2 + \lambda \sum_{j=1}^p \kappa_j \beta_j^2 \quad (28.1)$$

where $\lambda \geq 0$ is a tuning parameter that needs to be provided and p is the number of regressors. Different values of the penalty parameter λ lead to different ridge estimators. The regressors \mathbf{x} are standardized, and some simpler methods set $\kappa_j = 1$ for all j .

The first term in the objective function is the sum of squared residuals minimized by the OLS estimator. The second term is a penalty that for given λ is likely to increase with the number of regressors p .

The resulting ridge estimator when all $\kappa_j = 1$ can be expressed as

$$\hat{\beta}_\lambda = (\mathbf{X}' \mathbf{X} + \lambda N \mathbf{I}_p)^{-1} \mathbf{X}' \mathbf{y}$$

where p is the number of regressors. This estimator is a shrinkage estimator because it shrinks the OLS estimator $\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$, the special case $\lambda = 0$, toward $\mathbf{0}$. In the simplest case that the regressor matrix \mathbf{X} is orthonormalized so that $\mathbf{X}'\mathbf{X} = \mathbf{I}_p$, the OLS estimator is $\mathbf{X}'\mathbf{y}$, and the ridge estimator is $\mathbf{X}'\mathbf{y}/(1 + \lambda)$, which shrinks all coefficients toward 0 by the same multiplicative factor. For a given specification, a shrinkage factor that would lower the MSE of the prediction can be shown to exist; the practical task is to estimate it.

When the basic ridge regression is used, it is customary to standardize the regressors to have zero mean and unit variance. Some references and ridge regression programs assume that the dependent variable has been demeaned to have zero mean. In that case, y_i is replaced with $y_i - \bar{y}$, and, without loss of generality, the intercept can be dropped. The following code standardizes the regressors and demeans the dependent variable.

```
. * Standardize regressors and demean y
. foreach var of varlist x1 x2 x3 {
    2.      qui egen double z`var' = std(`var')
    3. }

. qui summarize y
. qui generate double ydemeaned = y - r(mean)
. summarize ydemeaned z*
```

Variable	Obs	Mean	Std. dev.	Min	Max
ydemeaned	40	-3.33e-17	3.400129	-6.650633	7.501798
zx1	40	2.63e-17	1	-1.594598	2.693921
zx2	40	2.62e-17	1	-2.34211	2.80662
zx3	40	-2.98e-17	1	-1.688912	2.764129

The Stata commands presented below for shrinkage estimation automatically standardize regressors. Then the preceding code is unnecessary.

There are several ways to choose the value of the penalty parameter λ . It can be determined by cv, and algorithms exist to quickly compute $\hat{\beta}_\lambda$ for many values of λ . Alternatively, penalty measures such as BIC may be used. Then the penalty based on the number of regressors p may be replaced by

the effective degrees of freedom, which for ridge regression can be shown to equal $\sum_{j=1}^p \lambda_j / (\lambda_j + \lambda)$, where λ_j are the eigenvalues of $\mathbf{X}'\mathbf{X}$.

28.3.2 Lasso

The lasso estimator, due to [Tibshirani \(1999\)](#), reduces MSE by setting some coefficients to zero. Additionally, the coefficients of retained variables are shrunk toward zero. Unlike ridge regression, the lasso can be used for variable selection.

The lasso estimator $\tilde{\boldsymbol{\beta}}_\lambda$ of $\boldsymbol{\beta}$ minimizes

$$Q_\lambda(\boldsymbol{\beta}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{x}'_i \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^p \kappa_j |\beta_j| \quad (28.2)$$

where $\lambda \geq 0$ is a tuning parameter that needs to be provided and p is the number of potential regressors. Different values of the penalty parameter λ lead to different lasso estimators. The regressors \mathbf{x} are standardized, and some simpler implementations set $\kappa_j = 1$ for all j .

The first term in the objective function is the sum of squared residuals minimized by the OLS estimator. The second term is a penalty measure based on the absolute value of the parameters, unlike the ridge estimator, which uses the square of the parameters.

There is no explicit solution for the resulting lasso estimator. It can be shown that the lasso estimator sets all but $k \leq p$ of the β_j coefficients to zero, where k is a decreasing function of λ , while also shrinking nonzero coefficients toward zero.

To see that some β_j may equal zero, suppose there are two regressors. The combinations of β_1 and β_2 for which the sum of squared residuals $\sum_{i=1}^N (y_i - \beta_1 x_{1i} - \beta_2 x_{2i})^2$ is constant define an ellipse. Different values of the sum of squared residuals correspond to different ellipses, given in

figure 28.1, and the OLS estimator is the centroid of the ellipses. In general, the lasso can be shown to equivalently minimize $\sum_{i=1}^N (y_i - \mathbf{x}'_i \boldsymbol{\beta})^2$ subject to the constraint that $\sum_{j=1}^p \kappa_j |\beta_j| \leq s$, where higher values of s correspond to lower values of λ . Specializing to the case $p = 2$, and letting $\kappa_1 = \kappa_2 = 1$, we see the lasso constraint $|\beta_1| + |\beta_2| \leq s$ defines the diamond-shaped region in the left panel of figure 28.1, and we are likely to wind up at one of the corners where $\beta_1 = 0$ or $\beta_2 = 0$. By contrast, the ridge estimator constraint $\beta_1^2 + \beta_2^2 \leq s$ defines a circle, and a corner solution is very unlikely.

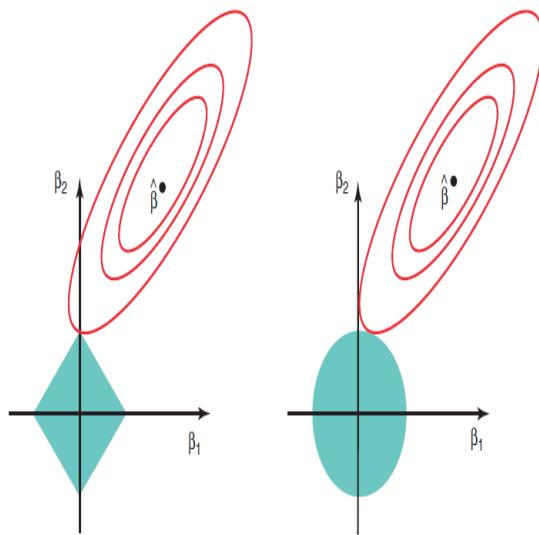


Figure 28.1. Lasso versus ridge

Note that lasso picks the best-fitting linear combination $\mathbf{x}'\boldsymbol{\beta}$ subject to the lasso constraint, rather than the best variables \mathbf{x} . This is especially the case when variables are correlated, such as when potential regressors include powers and interactions of underlying variables. Thus, the exact variables selected will vary in repeated samples or if there is a different partition of the data into K folds.

For prediction, lasso works best when a few of the potential regressors have $\beta_j \neq 0$, while most $\beta_j = 0$. By comparison, ridge regression works best when many predictors are important and have coefficients of

standardized regressors that are of similar size. The lasso is suited to variable selection, whereas ridge regression is not.

[Hastie, Tibshirani, and Friedman \(2009, chap. 3.8\)](#) discuss several penalized or regularized estimators that can be viewed as variations of the lasso, including the grouped lasso, the smoothly clipped absolute deviation penalty, and the Dantzig selector. The lasso estimator is a special case of a more general method called least-angle regression. The community-contributed `lars` command ([Mander 2006](#)) implements least-angle regression; the `a(lasso)` option obtains lasso estimates.

A thresholded or relaxed lasso performs an additional modified lasso using only those variables chosen by the initial lasso. The adaptive lasso presented in section [28.4.4](#) is an example.

28.3.3 Elastic net

In many applications, variables can be highly correlated with each other. The lasso penalty will drop many of these correlated variables, while the ridge penalty shrinks the coefficients of correlated variables toward each other.

The elastic net combines ridge regression and lasso. This can improve the MSE, but will retain more variables than the lasso. The `elasticnet` command has objective function

$$Q_{\lambda, \alpha}(\boldsymbol{\beta}) = \frac{1}{2N} \sum_{i=1}^N (y_i - \mathbf{x}'_i \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^p \kappa_j \left\{ \alpha |\beta_j| + \frac{1-\alpha}{2} \beta_j^2 \right\} \quad (28.3)$$

The ridge penalty averages correlated variables, while the lasso penalty leads to sparsity. Ridge is the special case $\alpha = 0$, and lasso is the special case $\alpha = 1$.

28.3.4 Finite sample distribution of lasso-related estimators

A model-selection method is consistent if asymptotically it correctly selects the correct model from a selection of candidate models. A model-selection method is conservative if asymptotically it always selects a model that nests the correct model. Selecting a model on the basis of minimum BIC is a consistent model-selection procedure, while selecting a model on the basis of minimum AIC is conservative ([Leeb and Pötscher 2005](#)).

A statistical model-selection and estimation method is said to have an oracle property if it leads to consistent model selection and a subsequent estimator that is asymptotically equivalent to the estimator that could be obtained if the true model was known so that model selection was unnecessary.

For example, suppose $y_i = \alpha x_{1i} + \beta x_{2i} + u_i$ and the true model is one with either $\beta = 0$ or $\beta \neq 0$. A consistent model-selection method correctly determines whether $\beta = 0$. Let $\hat{\alpha}$ be the estimator of α that first uses a model-selection method to determine whether $\beta = 0$ and then estimates whichever model is selected. Then $\hat{\alpha}$ has the oracle property if its asymptotic distribution is the same as that for the infeasible estimator of α that directly fits the true model without initial model selection.

The lasso is a consistent model-selection procedure but does not have the oracle property because of its bias. The adaptive lasso presented in section [28.4.4](#) is one of several variations of lasso that does have the oracle property.

Unfortunately, the oracle property is an asymptotic property that, while potentially useful in some settings such as recognizing numbers on a license plate, does not carry over to the finite sample settings that economists encounter. Our models do not fit perfectly, and we expect that with more observations, we can detect more variables that predict the outcome of interest. [Leeb and Pötscher \(2005\)](#), for example, consider the preceding example where β is of order $O(1/\sqrt{N})$. Then even though asymptotically the oracle property may still hold, $\hat{\alpha}$ has a complicated finite sample distribution that is affected by the first-stage determination of whether $\beta = 0$. In fact, $\hat{\alpha}$ has MSE that can be very large and even larger than that if we simply estimated both α and β without first determining whether $\beta = 0$. Mathematically, this difference between finite sample and asymptotic

performance is due to the asymptotic convergence not being uniform with respect to parameters.

Thus, we cannot perform standard inference on lasso or postlasso OLS coefficient estimates. Instead, if inference on parameters is desired, some model structure is required, and more complicated estimation methods need to be used. These are presented in sections [28.8](#) and [28.9](#).

28.4 Prediction using lasso, ridge, and elasticnet

We present an application of prediction for linear models using the lasso and the related shrinkage estimators—ridge and elastic net. These methods can be adapted to binary outcome models (logit and probit) and exponential mean models (Poisson), and we provide a logit example.

28.4.1 The lasso command

Lasso estimates can be obtained using the `lasso` command, which has syntax

```
lasso model depvar [ (alwaysvars) ] othervars [ if ] [ in ] [ weight ] [ , options ]
```

The model is one of `linear`, `logit`, `probit`, or `poisson`. The variables *alwaysvars* are variables to be always included, while the lasso selects among the *othervars* variables. The penalty λ can be determined by `cv` (option `selection(cv)`), adaptive `cv` (option `selection(adaptive)`), `BIC` (option `selection(bic)`), or a plugin formula (option `selection(plugin)`). For `cv`, the options include `folds(#)` for the number of folds. The plugin methods are intended for nonprediction use of the lasso; see section [28.8](#). Other options set tolerances for optimization.

The `lasso` command actually uses $1/2N$ rather than $1/N$ in [\(28.2\)](#). For clustered data, the option `cluster(clustvar)` defines the objective function to be the average over clusters of the within-cluster sums. Then [\(28.2\)](#) becomes

$$Q_\lambda(\boldsymbol{\beta}) = \frac{1}{G} \sum_{g=1}^G \left\{ \frac{1}{N_g} \sum_{i=1}^{N_g} (y_{ig} - \mathbf{x}'_{ig}\boldsymbol{\beta})^2 \right\} + \lambda \sum_{j=1}^{N_g} \kappa_j |\beta_j|$$

`cv` then selects folds at the cluster level, which requires a considerable number of clusters.

The `lasso` command output focuses on determination of the penalty λ . Postestimation commands `lassoinfo`, `lassoknots`, `lassoselect`, `cvplot`, `lassocoef`, `coefpath`, `lassogof`, and `bicplot` provide additional information. These commands are illustrated below.

The `elasticnet` command has syntax, options, and postestimation commands similar to those for the `lasso` command. Ridge estimates can be obtained using the `elasticnet` command with option `alpha(0)`. Stata also includes a `sqrlasso` command, which is seldom used and is not covered here.

28.4.2 Lasso linear regression example

We apply the `lasso linear` command to the current data example. Because there are only 40 observations, we use 5-fold cv rather than the default of 10 folds. The five folds are determined by a random-number generator, so for replicability, we need to set the seed.

```

. * Lasso linear using 5-fold CV
. lasso linear y x1 x2 x3, selection(cv, folds(5)) rseed(10101)

5-fold cross-validation with 100 lambdas ...
Grid value 1:      lambda = 1.591525    no. of nonzero coef. =      0
Folds: 1...5      CVF = 11.85738
Grid value 2:      lambda = 1.450138    no. of nonzero coef. =      1
Folds: 1...5      CVF = 11.60145
Grid value 3:      lambda = 1.321312    no. of nonzero coef. =      1
Folds: 1...5      CVF = 11.2296
Grid value 4:      lambda = 1.20393     no. of nonzero coef. =      1
Folds: 1...5      CVF = 10.87719
Grid value 5:      lambda = 1.096976    no. of nonzero coef. =      1
Folds: 1...5      CVF = 10.60149
Grid value 6:      lambda = .9995238   no. of nonzero coef. =      1
Folds: 1...5      CVF = 10.38463
Grid value 7:      lambda = .9107289   no. of nonzero coef. =      1
Folds: 1...5      CVF = 10.20522
Grid value 8:      lambda = .8298222   no. of nonzero coef. =      1
Folds: 1...5      CVF = 10.05685
Grid value 9:      lambda = .7561031   no. of nonzero coef. =      1
Folds: 1...5      CVF = 9.934201
Grid value 10:     lambda = .688933    no. of nonzero coef. =      1
Folds: 1...5      CVF = 9.829713
Grid value 11:     lambda = .6277301   no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.739804
Grid value 12:     lambda = .5719643   no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.666469
Grid value 13:     lambda = .5211525   no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.606777
Grid value 14:     lambda = .4748548   no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.562824
Grid value 15:     lambda = .43267    no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.525748
Grid value 16:     lambda = .3942328   no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.493472
Grid value 17:     lambda = .3592102   no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.460115
Grid value 18:     lambda = .327299    no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.43311
Grid value 19:     lambda = .2982226   no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.411316
Grid value 20:     lambda = .2717294   no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.393794
Grid value 21:     lambda = .2475897   no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.393523
Grid value 22:     lambda = .2255945   no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.40661
Grid value 23:     lambda = .2055533   no. of nonzero coef. =      2
Folds: 1...5      CVF = 9.420332

```

```

Grid value 24: lambda = .1872925 no. of nonzero coef. = 2
Folds: 1...5 CVF = 9.434326
... cross-validation complete ... minimum found
Lasso linear model No. of obs = 40
No. of covariates = 3
Selection: Cross-validation No. of CV folds = 5

```

ID	Description	lambda	No. of nonzero coef.	Out-of- sample R-squared	CV mean prediction error
1	first lambda	1.591525	0	-0.0519	11.85738
20	lambda before	.2717294	2	0.1666	9.393794
* 21	selected lambda	.2475897	2	0.1666	9.393523
22	lambda after	.2255945	2	0.1655	9.40661
24	last lambda	.1872925	2	0.1630	9.434326

* lambda selected by cross-validation.

The default grid for λ is a decreasing logarithmic grid of 100 values with $\lambda_j = \lambda_1 \times 10^{-4(j-1)/99}$, $j = 2, \dots, 100$, where λ_1 is the smallest value at which no variables are selected. Here $\lambda_1 = 1.591525$, and, for example, $\lambda_2 = \lambda_1 \times 10^{-4/99} = 1.591525 \times 0.97700996 = 1.450138$.

The output shows that reducing the penalty to $\lambda_2 = 1.450$ led to the inclusion of a regressor and that reducing the penalty to $\lambda_{11} = 0.628$ led to the inclusion of a second regressor. The cv objective function continued to decline to a minimum value of 9.394 at $\lambda_{21} = 0.248$. The results are listed only to the 24th largest grid value of λ , rather than all 100 grid-point values, because the minimum cv value has already been attained by then.

28.4.3 Lasso postestimation commands example

The `lassoknots` command provides a summary of the values of λ at which variables are selected or deselected. Additionally, it lists which variables were selected or deselected.

```
. * List the values of lambda at which variables are added or removed
. lassoknots
```

ID	lambda	No. of nonzero coef.	CV mean pred. error	Variables (A)dded, (R)emoved, or left (U)nchanged
2	1.450138	1	11.60145	A x1
11	.6277301	2	9.739804	A x2
* 21	.2475897	2	9.393523	U
24	.1872925	2	9.434326	U

* lambda selected by cross-validation.

In this example, once a variable is selected, it remains selected. The option `alllambda` gives results for all knots, and the option `display()` can produce additional statistics at each listed knot such as the number of nonzero coefficients and BIC.

The `lassoselect` command enables specifying a particular value for the optimal value λ^* following lasso with the option `selection(cv)`. The command `lassoselect ID=11` will change the selected λ^* to that with `ID=11` (here $\lambda^* = 0.6277301$). And `lassoselect lambda=0.50` will set λ^* to the grid value closest to 0.50 (here $\lambda_{13} = 0.5211525$).

The `cvplot` command plots the cv objective function against λ on a logarithmic scale or reverse logarithmic scale (the default) or plots the cv objective function against $\sum_j |\hat{\beta}_j|$.

```
. * Plot the change in the penalized objective function as lambda changes
. cvplot
```

The plot is given in the first panel of figure [28.2](#) and shows that cv decreases as the penalty λ decreases until $\lambda = 0.248$, at which point cv begins to increase.

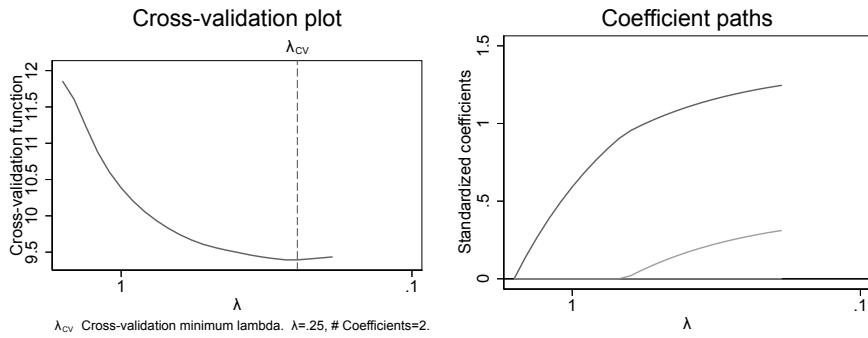


Figure 28.2. Plots of cv values and coefficients as λ changes

The `coefpath` command provides a similar plot for the standardized coefficients of each selected variable.

- . * Plot how estimated coefficients change with lambda
- . `coefpath, xunits(rlnlambda)`

The plot is given in the second panel of figure [28.2](#). In this example, once a variable is selected, it remains selected.

The `lassoinfo` command provides a summary of the lasso estimation.

- . * Provide a summary of the lasso
 - . `lassoinfo`
- ```
Estimate: active
Command: lasso
```

| Dependent variable | Model  | Selection method | Selection criterion | lambda   | No. of selected variables |
|--------------------|--------|------------------|---------------------|----------|---------------------------|
| y                  | linear | cv               | CV min.             | .2475897 | 2                         |

The `lassocoef` command with the `display(coef, )` option can provide three different sets of estimates for in-sample regression of  $y$  on the lasso-selected regressors.

Standardized coefficients (the default) are the estimates from lasso of  $y$  on the standardized regressors. These are the estimates directly obtained by the lasso at  $\lambda^*$ .

```
. * Lasso coefficients for the standardized regressors
. lassocoef, display(coef, standardized)
```

|       | active   |
|-------|----------|
| x1    | 1.206056 |
| x2    | .2715635 |
| _cons | 0        |

Legend:

- b - base level
- e - empty cell
- o - omitted

Penalized coefficients are the preceding standardized lasso estimates rescaled so that the standardization of variables is removed. These estimates can be interpreted in terms of the original data before standardization.

```
. * Lasso coefficients for the unstandardized regressors
. lassocoef, display(coef, penalized) nolegend
```

|       | active   |
|-------|----------|
| x1    | 1.35914  |
| x2    | .2918877 |
| _cons | 2.617622 |

The penalized coefficients are similar to the standardized coefficients because in this example the variables  $x_1$  and  $x_2$  had variances close to one.

Postselection coefficients are obtained by OLS regression of  $y$  on the lasso-selected regressors, here  $x_1$  and  $x_2$ . These are sometimes referred to as postlasso OLS estimates. [Belloni and Chernozhukov \(2013\)](#) find that the postlasso OLS estimator has lower bias and rate of convergence at least as good as the lasso estimator, and this is the case even if the lasso fails to include some relevant variables.

```
. * Postselection estimated coefficients for the unstandardized regressors
. lassocoef, display(coef, postselection) nolegend
```

|       | active   |
|-------|----------|
| x1    | 1.544198 |
| x2    | .4683922 |
| _cons | 2.533663 |

As expected, the lasso-penalized estimates of the coefficients of the selected variables (1.359 and 0.292) were smaller than these OLS estimates.

The `lassogof` command provides the goodness of fit with penalized coefficients (the default) or with postselection coefficients. We have

```
. * Goodness of fit with penalized coefficients and postselection coefficients
. lassogof, penalized
```

Penalized coefficients

| MSE      | R-squared | Obs |
|----------|-----------|-----|
| 8.679274 | 0.2300    | 40  |

```
. lassogof, postselection
```

Postselection coefficients

| MSE      | R-squared | Obs |
|----------|-----------|-----|
| 8.597958 | 0.2372    | 40  |

The postselection estimator is OLS, which maximizes  $R^2$  because it minimizes the sum of squared residuals. The lasso added a penalty, which necessarily leads to smaller in-sample  $R^2$ . The difference here between 0.2372 and 0.2300 is not great.

Finally, we verify that the postselection estimates are indeed obtained by OLS of  $y$  on the lasso-selected variables.

```
. * Compare with OLS with the lasso-selected regressors
. regress y x1 x2, noheader
```

| y     | Coefficient | Std. err. | t    | P> t  | [95% conf. interval] |
|-------|-------------|-----------|------|-------|----------------------|
| x1    | 1.544198    | .6305617  | 2.45 | 0.019 | .2665582 2.821837    |
| x2    | .4683922    | .6014166  | 0.78 | 0.441 | -.7501936 1.686978   |
| _cons | 2.533663    | .5159805  | 4.91 | 0.000 | 1.488188 3.579139    |

## 28.4.4 Adaptive lasso

The adaptive lasso ([Zou 2006](#)) is a multistep lasso method that usually leads to fewer variables being selected compared with the basic cv method.

The preceding analysis set  $\kappa_j = 1$  in [\(28.2\)](#). Adaptive lasso also begins with regular cv lasso (or cv ridge) with  $\kappa_j = 1$ . Adaptive lasso then does a second lasso that excludes variables with  $\hat{\beta}_j = 0$  and for the remainder sets  $\kappa_j = 1/|\hat{\beta}_j|^\delta$  with default  $\delta = 1$ , which favors variables with a larger coefficient because they receive a smaller penalty. The default is to have one adaptive step, but additional adaptive steps can be requested.

For the current example with one adaptive step, we obtain

```
. * Lasso linear using 5-fold adaptive CV
. qui lasso linear y x1 x2 x3, selection(adaptive, folds(5)) rseed(10101)
. lassoknobs
```

| ID   | lambda   | No. of nonzero coef. | CV mean pred. error | Variables (A)dded, (R)emoved, or left (U)nchanged |
|------|----------|----------------------|---------------------|---------------------------------------------------|
| 26   | 3.945214 | 1                    | 11.60145            | A x1                                              |
| * 52 | .3512089 | 1                    | 9.160539            | U                                                 |
| 57   | .2205694 | 2                    | 9.210699            | A x2                                              |
| 95   | .0064297 | 2                    | 9.172378            | U                                                 |

\* lambda selected by cross-validation in final adaptive step.

Now the optimal choice of  $\lambda$  leads to only  $x_1$  being selected.

The `selection(none)` option fits at each value of  $\lambda$  on the grid but does not select an optimal value of  $\lambda$ .

```
. qui lasso linear y x1 x2 x3, selection(none)
. lassoknnts
```

| ID | lambda   | No. of nonzero<br>coef. | In-sample<br>R-squared | Variables (A)dded, (R)emoved,<br>or left (U)nchanged |
|----|----------|-------------------------|------------------------|------------------------------------------------------|
| 2  | 1.450138 | 1                       | 0.0382                 | A x1                                                 |
| 11 | .6277301 | 2                       | 0.1908                 | A x2                                                 |
| 52 | .0138423 | 3                       | 0.2372                 | A x3                                                 |
| 62 | .0054597 | 3                       | 0.2373                 | U                                                    |

Note: No lambda selected. lassoselect can be used to select lambda.

For example, all three variables are selected if  $\lambda \leq 0.0138423$ .

The `selection(bic)` option uses the BIC, computationally faster than using `cv`, with the number of parameters set to the number of nonzero coefficients. The default is to evaluate the BIC at the penalized coefficients; the `selection(bic, postselection)` option instead evaluates at the postselection coefficients. The BIC is computed using a quasilikelihood function that assumes independence of observations, so care is needed in using it with clustered data.

The `selection(plugin)` option uses a plugin iterative procedure to determine  $\lambda^*$ . This option is intended for use in estimation, rather than prediction, and is presented in section [28.8](#).

#### 28.4.5 elasticnet command and ridge regression

The `elasticnet` command for ridge and elastic net estimation has syntax, options, and postestimation commands similar to those for the `lasso` command.

For the elastic net objective function given in [\(28.3\)](#), ridge regression is the special case  $\alpha = 0$ , and lasso is the special case  $\alpha = 1$ . Similarly, for the `elasticnet` command, the option `alpha(0)` implements ridge regression, and the option `alpha(1)` implements lasso.

We begin with ridge regression, using the option `alpha(0)`. Using five-fold `cv` to obtain the optimal  $\lambda$ , we have

```
. * Ridge estimation using the elasticnet command and selected results
. qui elasticnet linear y x1 x2 x3, alpha(0) rseed(10101) selection(cv, folds(5))
. lassoknots
```

| alpha | ID | lambda   | No. of nonzero<br>coef. | CV mean<br>pred.<br>error | Variables (A)dded, (R)emoved,<br>or left (U)nchanged |
|-------|----|----------|-------------------------|---------------------------|------------------------------------------------------|
| 0.000 | 1  | 1591.525 | 3                       | 11.9595                   | A x1 x2<br>x3                                        |
| * 93  |    | .3052401 | 3                       | 9.54017                   | U                                                    |
| 100   |    | .1591525 | 3                       | 9.566065                  | U                                                    |

\* alpha and lambda selected by cross-validation.

```
. lassocoef, display(coef, penalized) nolegend
```

|       | active   |
|-------|----------|
| x1    | 1.139476 |
| x2    | .4865453 |
| x3    | .0958546 |
| _cons | 2.659647 |

```
. lassogof, penalized
```

Penalized coefficients

| MSE     | R-squared | Obs |
|---------|-----------|-----|
| 8.70562 | 0.2277    | 40  |

The ridge coefficient estimates are on average shrunk toward 0 compared with the OLS slope estimates of, respectively, 1.555, 0.471, and – 0.026, given in section [28.2](#). And  $R^2$  has fallen from 0.2373 to 0.2277.

For elastic net regression, the `elasticnet` command performs a two-dimensional grid search over both  $\lambda$  and  $\alpha$ . The default for  $\lambda$  is the same logarithmic grid with 100 points as used by `lasso`, while  $\alpha = 0.5, 0.7, 1.0$ . For this example, the defaults led to  $\alpha = 1$ , so elastic net reduced to lasso. We specify a narrower grid that leads to  $\alpha = 0.95$ .

```

. * Elastic net estimation and selected results
. qui elasticnet linear y x1 x2 x3, alpha(0.9(0.05)1) rseed(10101)
> selection(cv, folds(5))
. lassoknots

```

| alpha | ID   | lambda   | No. of nonzero<br>coef. | CV mean<br>pred.<br>error | Variables (A)dded, (R)emoved,<br>or left (U)nchanged |
|-------|------|----------|-------------------------|---------------------------|------------------------------------------------------|
| 1.000 | 4    | 1.450138 | 1                       | 11.60145                  | A x1                                                 |
|       | 13   | .6277301 | 2                       | 9.739804                  | A x2                                                 |
|       | 26   | .1872925 | 2                       | 9.434326                  | U                                                    |
| 0.950 | 29   | 1.591525 | 1                       | 11.73019                  | A x1                                                 |
|       | 38   | .688933  | 2                       | 9.81611                   | A x2                                                 |
|       | * 48 | .2717294 | 2                       | 9.3884                    | U                                                    |
|       | 51   | .2055533 | 2                       | 9.425887                  | U                                                    |
| 0.900 | 53   | 1.675289 | 1                       | 11.74015                  | A x1                                                 |
|       | 62   | .7561031 | 2                       | 9.900317                  | A x2                                                 |
|       | 76   | .2055533 | 2                       | 9.431641                  | U                                                    |

\* alpha and lambda selected by cross-validation.

```
. lassocoef, display(coef, penalized) nolegend
```

|       | active   |
|-------|----------|
| x1    | 1.329744 |
| x2    | .2908281 |
| _cons | 2.627567 |

```
. lassogof, penalized
```

Penalized coefficients

| MSE      | R-squared | Obs |
|----------|-----------|-----|
| 8.693386 | 0.2288    | 40  |

The optimal values of  $\alpha = 0.95$  and  $\lambda = 0.2717$  lead to selection of  $x_1$  and  $x_2$ . The penalized coefficient estimates and MSE are close to the lasso estimates, the case  $\alpha = 1$ . In real-life examples with many more regressors, we expect a bigger difference.

## 28.4.6 In-sample comparison of shrinkage estimators

The results across several shrinkage model estimates can be compared using the postestimation commands `lassocoef`, `lassogof`, and `lassoinfo`.

First, save model results using the `estimates store` command.

```
. * Fit various models and store results
. qui regress y x1 x2 x3
. estimates store OLS
. qui lasso linear y x1 x2 x3, selection(cv, folds(5)) rseed(10101)
. estimates store LASCV
. qui lasso linear y x1 x2 x3, selection(adaptive, folds(5)) rseed(10101)
. estimates store LASADAPT
. qui lasso linear y x1 x2 x3, selection(plugin)
. estimates store LASPLUG
. qui elasticnet linear y x1 x2 x3, alpha(0) selection(cv, folds(5))
> rseed(10101)
. estimates store RIDGECV
. qui elasticnet linear y x1 x2 x3, alpha(0.9(0.05)1) rseed(10101)
> selection(cv, folds(5))
. estimates store ELASTIC
```

We compare in-sample model fit and the specific variables selected. The comparison below uses penalized coefficient estimates for standardized variables. For unpenalized postselection estimates of unstandardized variables, use `lassogof` option `postselection` and `lassocoef` option `display(coef, postselection)`.

```
. * Compare in-sample fit and selected coefficients of various models
. lassogof OLS LASCV LASADAPT LASPLUG RIDGECV ELASTIC
```

Penalized coefficients

| Name     | MSE      | R-squared | Obs |
|----------|----------|-----------|-----|
| OLS      | 8.597403 | 0.2373    | 40  |
| LASCV    | 8.679274 | 0.2300    | 40  |
| LASADAPT | 8.755573 | 0.2232    | 40  |
| LASPLUG  | 10.23264 | 0.0922    | 40  |
| RIDGECV  | 8.70562  | 0.2277    | 40  |
| ELASTIC  | 8.693386 | 0.2288    | 40  |

```
. lassocoef OLS LASCV LASADAPT LASPLUG RIDGECV ELASTIC, display(coef) nolegend
```

|       | OLS       | LASCV    | LASADAPT | LASPLUG  | RIDGECV  | ELASTIC  |
|-------|-----------|----------|----------|----------|----------|----------|
| x1    | 1.555582  | 1.206056 | 1.462431 | .3693423 | 1.011134 | 1.179972 |
| x2    | .4707111  | .2715635 |          |          | .452667  | .2705777 |
| x3    | -.0256025 |          |          |          | .0979251 |          |
| _cons | 2.531396  | 0        | 0        | 0        | 0        | 0        |

cv lasso selects  $x_1$  and  $x_2$ , while adaptive lasso, which provides a bigger penalty than cv lasso, selects only  $x_1$ . Ridge by construction retains all variables, while the elastic net in this example selected  $x_1$  and  $x_2$ .

All methods have similar in-sample MSE and  $R^2$ , aside from plugin lasso. If we had additionally made out-of-sample predictions, then we expect that the shrinkage estimators would predict better than OLS with regressors  $x_1$ ,  $x_2$ , and  $x_3$ .

#### 28.4.7 Shrinkage for logit, probit, and Poisson models

In principle, the lasso, ridge, and elastic net penalties can be applied to objective functions other than the sum of squared residuals used for linear regression.

In particular, for generalized linear models, the objective function uses the sum of squared deviance residuals, defined in section 13.8.3, rather than the sum of squared residuals. The `lasso` and `elasticnet` commands can also be applied to logit, probit, and Poisson models.

The squared residual  $(y_i - \mathbf{x}'_i \boldsymbol{\beta})^2$  in (28.1), (28.2), and (28.3) is replaced by the squared deviance residual. For logit, this term is  $2[y_i \ln \Lambda(\mathbf{x}'_i \boldsymbol{\beta}) + (1 - y_i) \ln\{1 - \Lambda(\mathbf{x}'_i \boldsymbol{\beta})\}]$ , where  $\Lambda(z) = e^z/(1 + e^z)$ . For probit, we use  $2[y_i \ln \Phi(\mathbf{x}'_i \boldsymbol{\beta}) + (1 - y_i) \ln\{1 - \Phi(\mathbf{x}'_i \boldsymbol{\beta})\}]$ , where  $\Phi(\cdot)$  is the standard normal cumulative distribution function. For Poisson, we use  $2\{y_i \mathbf{x}'_i \boldsymbol{\beta} - \exp(\mathbf{x}'_i \boldsymbol{\beta}) - v_i\}$ , where  $v_i = 0$  if  $y_i = 0$  and  $v_i = y_i \ln(y_i) - y_i$  otherwise.

The related Stata commands in the case of lasso are, respectively, `lasso logit`, `lasso probit`, and `lasso poisson`.

To illustrate the method for a binary variable, we convert `y` to a variable `dy` that takes value 1 if  $y > 3$  and implement lasso for a logit model with  $\lambda$  determined by five-fold cv.

```
. * Lasso for logit example
. qui generate dy = y > 3
. qui lasso logit dy x1 x2 x3, rseed(10101) selection(cv, folds(5))
. lassoknots
```

| ID   | lambda   | No. of nonzero<br>coef. | CV mean<br>deviance | Variables (A)dded, (R)emoved,<br>or left (U)nchanged |
|------|----------|-------------------------|---------------------|------------------------------------------------------|
| 2    | .2065674 | 1                       | 1.407613            | A x1                                                 |
| * 24 | .0266792 | 1                       | 1.192646            | U                                                    |
| 26   | .0221495 | 2                       | 1.192865            | A x2                                                 |
| 30   | .0152668 | 3                       | 1.194545            | A x3                                                 |
| 31   | .0139106 | 3                       | 1.195055            | U                                                    |

\* lambda selected by cross-validation.

The optimal  $\lambda$  leads to selection of only `x1`.

For a count example, we create a Poisson variable `ycount` that takes values between 0 and 7 and whose mean depends on only `x1`. lasso with five-fold cv yields

```

. * Lasso for count data example
. qui generate ycount = rpoisson(exp(-1 + x1))
. qui lasso poisson ycount x1 x2 x3, rseed(10101) selection(cv, folds(5))
. lassoknots

```

| ID   | lambda   | No. of<br>nonzero<br>coef. | CV mean<br>deviance | Variables (A)dded, (R)emoved,<br>or left (U)nchanged |
|------|----------|----------------------------|---------------------|------------------------------------------------------|
| 2    | 1.012329 | 1                          | 2.191141            | A x1                                                 |
| * 25 | .119132  | 1                          | .8257619            | U                                                    |
| 29   | .0821131 | 2                          | .8334985            | A x3                                                 |

\* lambda selected by cross-validation.

Again, the optimal  $\lambda$  leads to selection of only  $x_1$ .

## 28.5 Dimension reduction

Dimension-reduction methods reduce the number of regressors from  $p$  to  $m < p$  linear combinations of regressors. Thus, given initial model  $\mathbf{y} = \beta_0 + \mathbf{X}\beta + \mathbf{u}$ , where  $\mathbf{X}$  is  $N \times p$ , we form matrix  $\mathbf{Z} = \mathbf{XA}$ , where  $\mathbf{A}$  is  $p \times m$  and  $\mathbf{Z}$  is  $N \times m$ . Then, we fit the model  $\mathbf{y} = \gamma_0 + \mathbf{Z}\gamma + \mathbf{v}$ .

Here we present principal components, a long-standing method that uses only  $\mathbf{X}$  to form  $\mathbf{A}$  (unsupervised learning). A related method is partial least squares, which additionally uses the relationship between  $\mathbf{y}$  and  $\mathbf{X}$  to form  $\mathbf{A}$  (supervised learning). Principal components is the method most often used in econometrics studies and can be used in a very wide range of applications.

### 28.5.1 Principal components

The principal components method selects the linear combinations of regressors, called principal components, as follows. The first principal component has the largest sample variance among all normalized linear combinations of the columns of  $\mathbf{X}$ . The second principal component has the largest sample variance subject to being orthogonal to the first, and so on. More formally the  $j$ th principal component is the  $N \times 1$  vector  $\mathbf{X}\mathbf{h}_j$ , where  $\mathbf{h}_j$  is the eigenvector corresponding to  $\lambda_j$ , the  $j$ th largest eigenvalue of  $\mathbf{X}'\mathbf{X}$ .

The principal components are not invariant to the scaling of  $\mathbf{X}$ , and it is common practice to apply principal components to data that have been standardized to have mean zero and variance one. Let  $\mathbf{X}^*$  denote the regressor matrix after this standardization. The Stata `pca` command computes the principal components. The default option is the `correlation` option that is equivalent to automatically standardizing the data before analysis. So with this default option, there is no need to first standardize the regressors. We obtain

```
. * Principal components using default option that first standardizes the data
. pca x1 x2 x3
```

| Principal components/correlation  | Number of obs   | =          | 40                    |
|-----------------------------------|-----------------|------------|-----------------------|
|                                   | Number of comp. | =          | 3                     |
|                                   | Trace           | =          | 3                     |
| Rotation: (unrotated = principal) | Rho             | =          | 1.0000                |
| <hr/>                             |                 |            |                       |
| Component                         | Eigenvalue      | Difference | Proportion Cumulative |
| Comp1                             | 1.81668         | 1.08919    | 0.6056 0.6056         |
| Comp2                             | .727486         | .27165     | 0.2425 0.8481         |
| Comp3                             | .455836         | .          | 0.1519 1.0000         |

Principal components (eigenvectors)

| Variable | Comp1  | Comp2   | Comp3   | Unexplained |
|----------|--------|---------|---------|-------------|
| x1       | 0.6306 | -0.1063 | -0.7688 | 0           |
| x2       | 0.5712 | -0.6070 | 0.5525  | 0           |
| x3       | 0.5254 | 0.7876  | 0.3220  | 0           |

The output includes the three eigenvalues and three eigenvectors. The data are standardized automatically, so each of the three variables has variance one, and the sum of the variances is three. The variance of each principal component equals the corresponding eigenvalue, so the first principal component has variance  $1.81668$  and explains a fraction  $1.81668/3 = 0.6056$  of the total variance.

Note that the same results are obtained if we use the previously standardized variables  $zx1-zx3$  and the `covariance` option, specifically, command `pca zx1 zx2 zx3, covariance`.

The `predict` postestimation command constructs variables equal to the three principal components. We obtain

```
. * Compute the three principal components and their means, st.devs., correlations
. predict pc1 pc2 pc3
(score assumed)
```

Scoring coefficients

sum of squares(column-loading) = 1

| Variable | Comp1  | Comp2   | Comp3   |
|----------|--------|---------|---------|
| zx1      | 0.6306 | -0.1063 | -0.7688 |
| zx2      | 0.5712 | -0.6070 | 0.5525  |
| zx3      | 0.5254 | 0.7876  | 0.3220  |

```
. summarize pc1 pc2 pc3
```

| Variable | Obs | Mean      | Std. dev. | Min       | Max      |
|----------|-----|-----------|-----------|-----------|----------|
| pc1      | 40  | -3.35e-09 | 1.347842  | -2.52927  | 2.925341 |
| pc2      | 40  | -3.63e-09 | .8529281  | -1.854475 | 1.98207  |
| pc3      | 40  | 2.08e-09  | .6751564  | -1.504279 | 1.520466 |

```
. correlate pc1 pc2 pc3
(obs=40)
```

|     | pc1     | pc2     | pc3    |
|-----|---------|---------|--------|
| pc1 | 1.0000  |         |        |
| pc2 | 0.0000  | 1.0000  |        |
| pc3 | -0.0000 | -0.0000 | 1.0000 |

The principal components have mean zero, standard deviation equal to the square root of the corresponding eigenvalue, for example,  
 $\sqrt{1.81668} = 1.3478$ , and are uncorrelated.

The principal components are computed applying the relevant eigenvectors to the standardized variables. For example, the first principal component is computed as follows:

```
. * Manually compute the first principal component and compare to pc1
. generate double pc1manual = 0.6306*zx1 + 0.5712*zx2 + 0.5254*zx3
. summarize pc1 pc1manual
```

| Variable  | Obs | Mean      | Std. dev. | Min       | Max      |
|-----------|-----|-----------|-----------|-----------|----------|
| pc1       | 40  | -3.35e-09 | 1.347842  | -2.52927  | 2.925341 |
| pc1manual | 40  | -9.02e-18 | 1.347822  | -2.529204 | 2.925356 |

The principal components are obtained without any consideration of regression on a variable  $y$ . If we regress  $y$  on all  $p$  principal components, we necessarily get the same predicted values of  $y$  and the same  $R^2$  as if we

regress  $y$  on all the original  $p$  regressors. The hope is that if we regress  $y$  on just, say, the first  $m < p$  principal components, then we obtain fit better than that obtained by arbitrarily picking  $m$  regressors and not much worse than if we used all  $p$  regressors. There is no guarantee this will happen, but in practice it often does.

The following example gives correlations of the dependent variable with fitted values from regression on, respectively, all three regressors, the first principal component,  $x_1$ ,  $x_2$ , and  $x_3$ . Recall that the square of these correlations equals  $R^2$  from the corresponding OLS regression.

```
. * Compare R from OLS on all three regressors, on pc1, on x1, on x2, on x3
. qui regress y x1 x2 x3
. predict yhat
(option xb assumed; fitted values)
. correlate y yhat pc1 x1 x2 x3
(obs=40)
```

|      | y      | yhat   | pc1    | x1     | x2     | x3     |
|------|--------|--------|--------|--------|--------|--------|
| y    | 1.0000 |        |        |        |        |        |
| yhat | 0.4871 | 1.0000 |        |        |        |        |
| pc1  | 0.4444 | 0.9122 | 1.0000 |        |        |        |
| x1   | 0.4740 | 0.9732 | 0.8499 | 1.0000 |        |        |
| x2   | 0.3370 | 0.6919 | 0.7700 | 0.5077 | 1.0000 |        |
| x3   | 0.2046 | 0.4200 | 0.7082 | 0.4281 | 0.2786 | 1.0000 |

There is some loss in fit because of using only the first principal component. The correlation has fallen from 0.4871 to 0.4444, corresponding to a fall in  $R^2$  from 0.237 to 0.197. Regression on  $x_1$  alone has better fit, as expected because the DGP in this example depended on  $x_1$  alone, while regressions on  $x_2$  alone and on  $x_3$  alone do not fit nearly as well as regression on the first principal component.

## 28.6 Machine learning methods for prediction

Machine learning methods are algorithms that determine a predictor using only the data at hand, rather than by using a researcher-specified model. The terms “machine learning”, “statistical learning”, and “data science” are to some extent interchangeable.

The lasso and other shrinkage estimators are leading examples of methods used in machine learning. In this section, we present additional machine learning methods.

The machine learning literature distinguishes between supervised learning, where an outcome variable  $y$  is observed, and unsupervised learning, where no outcome variable is observed. Within supervised learning, distinction is made between an outcome measured on a cardinal scale, most often continuous, and an outcome that is categorical. The latter case is referred to as classification.

Machine learning methods are often applied to big data, where the term “big data” can mean either many observations or many variables. It includes the case where the number of variables exceeds the number of observations, even if there are relatively few observations.

By allowing potential regressors to include powers and interactions of underlying variables, a linear (in parameters) model used by shrinkage estimators such as the lasso may actually explain the outcome sufficiently well. Other machine learning methods, such as neural networks and regression trees, do not transform the underlying variables but instead fit models that can be very nonlinear in these underlying variables.

The following overview summarizes some additional methods for prediction, many from the machine learning literature. Some of these methods are illustrated in the subsequent application section. The presentation is very dense and more advanced than much of the other material in this book. Little detail is provided on these methods, such as determination of necessary tuning parameters akin to  $\lambda$  for the lasso, though see chapter [27](#) for nonparametric and semiparametric methods. For more

details see, for example, [James et al. \(2021\)](#) or [Hastie, Tibshirani, and Friedman \(2009\)](#).

### 28.6.1 Supervised learning for continuous outcome

We have continuous outcome  $y$  that, given predictors  $\mathbf{x}$ , is predicted by function  $g(\mathbf{x})$ . OLS uses  $g(\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta}$  or, more precisely,  $g(\mathbf{x}) = \mathbf{z}'\boldsymbol{\beta}$ , where the regressors  $\mathbf{z}$  are specified functions of  $\mathbf{x}$  such as transformations and interactions. For simplicity, we do not distinguish between the underlying variables  $\mathbf{x}$  and the regressors  $\mathbf{z}$  formed from  $\mathbf{x}$ .

A quite general model for  $g(\mathbf{x})$  is a fully nonparametric model such as kernel regression or local polynomial regression, with the function  $g(\cdot)$  unspecified. Such models can be fit using the `npregress` command; see section [27.2.5](#). But this yields an imprecise estimate of  $g(\cdot)$  for high-dimensional  $\mathbf{x}$ , a problem referred to as the curse of dimensionality, and the method is not suited to prediction outside the domain of  $\mathbf{x}$ .

The econometrics literature has sought to overcome the curse of dimensionality by fitting semiparametric models that reduce the dimensionality of the nonparametric component, enabling estimation and inference on the parametric component. The leading examples—partial linear, single-index, and generalized additive models—were presented in sections [27.6–27.8](#). These semiparametric models are used to obtain estimates of parameters or partial effects, rather than for prediction per se. In section [28.8](#), we present estimation of key parameters in a partial linear model using lasso to select control variables.

### 28.6.2 Neural networks

Neural networks lead to quite flexible nonlinear models for  $g(\mathbf{x})$ . These models introduce a series of hidden layers between the outcome  $y$  and the regressors  $\mathbf{x}$ . Deep learning methods use neural networks.

For example, a neural network with two layers introduces an intermediate layer between input variables  $\mathbf{x}$  and the output  $y$ . The intermediate layer is composed of  $M$  intermediate units or hidden variables

$z_m, m = 1, \dots, M$ , that are each a nonlinear transformation of a linear combination of the inputs  $\mathbf{x}$ , so  $z_m = g(\alpha_{0m} + \mathbf{x}'\boldsymbol{\alpha}_m)$  for specified function  $g(\cdot)$ .

Initial research often used the sigmoid function  $g(v) = 1/(1 + e^{-v})$ . More recently, it is common to use rectified linear units with  $g(v) = \max(0, v)$ . The output is then a linear combination of the  $M$  hidden units, or a transformation of this linear combination, so  $E(y|\mathbf{x}) = h(\mathbf{t})$ , where  $\mathbf{t} = \beta_0 + \mathbf{z}'\boldsymbol{\beta}$  and usually  $h(\mathbf{t}) = \mathbf{t}$ . Given  $g(v) = 1/(1 + e^{-v})$  and  $h(\mathbf{t}) = \mathbf{t}$ , a two-layer neural network reduces to the nonlinear model  $E(y|\mathbf{x}) = \beta_0 + \sum_{m=1}^M \beta_j / \{1 + e^{-(\alpha_{0m} + \mathbf{x}'\boldsymbol{\alpha}_m)}\}$ . If MSE is the loss function, then estimation of the various  $\alpha$  and  $\beta$  parameters is a nonlinear least squares problem.

More complicated neural net models add additional hidden layers. There is a tendency to overfit, and a ridge-regression-type penalty may be used. There is an art to estimation of neural network models because they entail several tuning parameters—the number of layers, the number of hidden variables in each layer, the function  $g(\cdot)$ , and a penalty term for overfitting.

Neural network models are highly nonlinear. Estimation uses stochastic gradient descent, a variation of gradient-based methods that at each step calculates the gradient at a randomly chosen subset of the observations.

### 28.6.3 Regression trees

Regression trees sequentially split regressors  $\mathbf{x}$  into regions that best predict  $y$ . The prediction of  $y$  at a given value  $\mathbf{x}_0$  that falls in a region  $R^*$  is then the average of  $y$  across all observations for which  $\mathbf{x} \in R^*$ . This is equivalent to regression of  $y$  on a set of mutually exclusive indicator variables where each indicator variable corresponds to a given region of  $\mathbf{x}$  values.

Suppose we first split on the  $j$ th variable  $x_j$  at point  $s$ . Defining regions  $R1(j, s) = \{\mathbf{x} | x_j < s\}$  and  $R2(j, s) = \{\mathbf{x} | x_j \geq s\}$ , the MSE is  $(1/N) \sum_{i:\mathbf{x}_i \in R1(j,s)}^N (y_i - \bar{y}_{R1})^2 + (1/N) \sum_{i:\mathbf{x}_i \in R2(j,s)}^N (y_i - \bar{y}_{R2})^2$ . The

first split is based on a search over regressors  $x_j, j = 1, \dots, p$  and split points  $s$  to obtain  $(j, s)$  that minimizes this MSE. We then next search over possible splits of  $R1$  and  $R2$ , with possible split on any of the  $p$  regressors, and choose the additional split that minimizes MSE, and so on.

After  $K$  splits, the MSE is  $(1/N) \sum_{k=1}^K \sum_{i:\mathbf{x}_i \in R_k} (y_i - \bar{y}_{R_k})^2$ , where  $R_k$  denotes the  $k$ th terminal node. The prediction for  $\mathbf{x}_0 \in R_k$  is then the average of  $y$  over all  $\mathbf{x}_i \in R_k$ . So

$$\hat{g}(\mathbf{x}_0) = \{\sum_{k=1}^K \sum_{i:\mathbf{x}_i \in R_k} \mathbf{1}(\mathbf{x} \in R_k) y_i\} / \{\sum_{k=1}^K \sum_{i:\mathbf{x}_i \in R_k} \mathbf{1}(\mathbf{x}_i \in R_k)\},$$

where  $\mathbf{1}(A)$  is an indicator function equal to one if event  $A$  occurs and equal to zero otherwise.

Implementation requires specification of the depth of the tree and the minimum number of observations in the terminal nodes of the tree. The method takes a so-called greedy approach that determines the best split at each step without looking ahead and picking a split that could lead to a better tree in some future step. Thus, changes in the residual sum of squares is not used as a stopping criterion because better splits may still be possible. Instead, it is best to overfit with more splits than may be ideal and then prune back using a penalty function such as  $\lambda|T|$ , where  $|T|$  is the number of terminal nodes.

Simple regression trees have the advantage of interpretability if there are few regressors. However, predictions from a single regression tree have high variance. For example, splitting the sample into two can lead to two quite different trees.

#### 28.6.4 Bagging

Bagging and boosting are general methods for improving prediction that work especially well for regression trees.

Bagging, a shortening of “bootstrap aggregating”, reduces prediction variance by obtaining predictions for several different samples and averaging these predictions. The different samples are obtained by bootstrap that randomly chooses  $N$  observations with replacement from the original sample of  $N$  observations.

Specifically, for each of the  $b = 1, \dots, B$  bootstrap samples, we obtain a large tree and prediction  $\hat{g}_b(\mathbf{x})$  and then use the average prediction  $\hat{g}_{bag}(\mathbf{x}) = (1/B) \sum_{b=1}^B \hat{g}_b(\mathbf{x})$ . Because sampling is with replacement, some observations will appear in the bootstrap multiple times, while others will not appear at all. The observations not in a bootstrap sample can be used as a test sample—this replaces CV.

### 28.6.5 Random forests

The  $B$  bagging estimates will be correlated because the bootstrap samples have considerable overlap. This is especially the case for regression trees because if a regressor is especially important, it will appear near the top of the tree in every bootstrap sample.

A random forest adjusts bagging for regression trees as follows: within each bootstrap sample, each time a split is considered, only a random sample of  $m < p$  predictors is used in deciding the next split. Compared with a single regression tree, this adds  $m$  as an additional tuning parameter; often,  $m$  is set to the first integer greater than  $\sqrt{p}$ .

Random forests are related to kernel and  $k$ -nearest neighbors because they use a weighted average of nearby observations. Random forests can predict better because they have a data-driven way of determining which nearby observations get weight; see [Lin and Jeon \(2012\)](#).

### 28.6.6 Boosting

Boosting methods construct multiple predictions from reweighted data using the original sample, rather than by bootstrap resampling, and use as predictor a combination of these predictions. There are many boosting algorithms.

A common boosting method for regression trees for continuous outcomes sequentially updates the initial tree by applying a regression tree to residuals obtained from the previous stage. Specifically, given the  $b$ th stage model with predictions  $\hat{g}^b(\mathbf{x})$ , fit a decision tree  $\hat{h}^b(\mathbf{x})$  to residuals  $r^b$ , defined below, rather than to the outcome  $y$ . Then, update

$\widehat{g}^{b+1}(\mathbf{x}) = \widehat{g}^b(\mathbf{x}) + \lambda \widehat{h}^b(\mathbf{x})$ , where  $\lambda$  is a penalty parameter, and update the residuals  $r^{b+1} = r^b - \lambda \widehat{h}^b(\mathbf{x})$ . The boosted prediction is  
 $\widehat{g}_{\text{boost}}(\mathbf{x}) = (1/B) \sum_{b=1}^B \widehat{g}_b(\mathbf{x})$ .

### 28.6.7 Supervised learning for categorical outcome (classification)

Digital license plate recognition provides an example of categorical classification. Given a digital image of a number or letter, we aim to correctly categorize it.

With  $K$  categories, let  $y$  take values  $1, 2, \dots, K$ . The standard loss function used is the error rate that counts up the number of wrong classifications. Then,

$$\text{Error rate} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}(\widehat{y}_i \neq y_i)$$

where the indicator function  $\mathbf{1}(A) = 1$  if event  $A$  happens and equals 0 otherwise.

One method to predict  $y$  is to apply a standard parametric model for categorical data such as binary logit in the case of two categories, or more generally, multinomial logit, that yields predicted probabilities of being in each category. Then, allocate the  $i$ th observation to the category with the highest predicted probability. In the case of binary logit with outcome  $y$  taking values 1 or 2, we let  $\widehat{y}_i = 2$  if the predicted probability  $\widehat{P}(y_i = 2) > 0.5$  and let  $\widehat{y}_i = 1$  otherwise.

The following methods are felt to lead to classification with lower error rate than methods based on directly modeling  $\Pr(y = k | \mathbf{x})$ .

Discriminant analysis specifies a joint distribution for  $(y, \mathbf{x})$ . For linear discriminant analysis with  $K$  categories, in the  $k$ th category, we suppose  $\mathbf{x}|y = k \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$  and define  $\pi_k = \Pr(y = k)$ . Then, we obtain an

expression for  $\Pr(y = k|\mathbf{x})$  from Bayes theorem, evaluate this at sample estimates for  $\boldsymbol{\mu}_k$ ,  $\boldsymbol{\Sigma}$ , and  $\pi_k$ ,  $k = 1, 2, \dots, K$ , and assign the  $i$ th observation to category  $k$  with the largest estimated  $\Pr(y_i = k|\mathbf{x}_i)$ . The procedure is called linear discriminant analysis because the resulting classification rule can be shown to be a linear function of  $\mathbf{x}$ .

Quadratic discriminant analysis amends linear discriminant analysis by supposing  $\mathbf{x}|y = k \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ , so additionally the variance of  $\mathbf{x}$  varies across categories. The procedure is called quadratic discriminant analysis because the resulting classification rule can be shown to be a quadratic function of  $\mathbf{x}$ .

The preceding classifiers are restrictive because they define a boundary that is linear or quadratic. For example, if  $K = 2$ , then a linear classifier predicts  $y = 2$  according to whether  $\mathbf{x}'\mathbf{a} > b$  for model-determined coefficients  $\mathbf{a}$  and  $b$ . This rules out more flexible classifiers such as predicting  $y = 2$  if  $\mathbf{x}$  lies in a closed region and predicting  $y = 1$  if  $\mathbf{x}$  lies outside this closed region. A support vector machine allows such nonlinear boundaries; leading examples use what is called a polynomial kernel or a radial kernel.

The `discrim` command includes linear, quadratic, and  $k$ -nearest-neighbor discriminant analysis. For support vector machines, see the community-contributed `svmachines` command ([Guenther and Schonlau 2016](#)).

## 28.6.8 Unsupervised learning (cluster analysis)

In unsupervised learning, there is no observed outcome  $y$ , only predictors  $\mathbf{x}$ . The goal is to form one or more groups or clusters for which the predictors  $\mathbf{x}$  take similar values. An example is determining several types of individual personalities on the basis of a range of psychometric measures.

Principal components, introduced in section [28.5.1](#), provide one method. Then the first principal component defines the first group, the second principal component defines the second group, and so on.

$K$ -means clustering forms  $K$  distinct clusters for which the sum of within-cluster variation is minimized. A common measure of variation is Euclidean distance, in which case we choose clusters  $C_1, \dots, C_K$  to minimize  $\sum_{k=1}^K W(C_k)$ , where  $W(C_k) = (1/N_k) \sum_{i,i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2$  and  $N_k$  is the number of observations in cluster  $C_k$ .

Hierarchical clustering methods are sequential methods that start with many clusters that are combined to form fewer clusters, or that start with one cluster and then split clusters to form more clusters, until an optimal number of clusters is obtained.

The `cluster kmeans` command implements  $K$ -means clustering for continuous, binary, and mixed data using many different measures of distance.

## 28.7 Prediction application

We compare various methods of prediction using the chapter 3 data on natural logarithm of health expenditures. Several of the methods illustrated employ with little explanation community-contributed programs whose use requires additional reading.

### 28.7.1 Training and holdout samples

The sample is split into two parts. A training sample is used to select and estimate the preferred predictor within a class of predictors, such as lasso. A test sample or hold-out sample of the remaining observations is then used to compare the out-of-sample predictive ability of these various predictors.

The basic variables are 5 continuous variables and 14 binary variables. From these, we can create 188 interacted variables—20 from continuous variables and their own second-order interactions, 28 from the 14 binary variables, and 140 from the interactions of the binary and continuous variables.

```
. * Data for prediction example: 5 continuous and 14 binary variables
. qui use mus203mepsmedexp, clear
. keep if !missing(ltotexp)
(109 observations deleted)
. global xlist income educyr age famsze totchr
. global dlist suppins female white hisp marry northe mwest south
> msa phylim actlim injury priolist hvgg
. global rlist c.($xlist)##c.($xlist) i.($dlist) c.($xlist)#i.($dlist)
```

The sample is split into a training and fitting sample that uses 80% of the observations (`train==1`) and a holdout sample of 20% of the observations (`train==0`).

```
. splitsample ltotexp, generate(train) split(1 4) values(0 1) rseed(10101)
. tabulate train
```

| train | Freq. | Percent | Cum.   |
|-------|-------|---------|--------|
| 0     | 591   | 20.00   | 20.00  |
| 1     | 2,364 | 80.00   | 100.00 |
| Total | 2,955 | 100.00  |        |

Note that here the sample is completely observed because a preceding command dropped observations with missing values of `ltotexp`, the only variable with missing values. So including a list of variables such as `ltotexp` in `split` command is actually unnecessary.

## 28.7.2 Various predictors

We obtain estimates using only the training sample (`train==1`). The subsequent `predict` command provides predictions for the entire sample, so it provides predictions for both `train==1` and `train==0`.

The first predictor is obtained by OLS regression of `ltotexp` on the 19 basic variables using the training sample. Most variables are statistically significant at the 5% level.

```
. * OLS with 19 regressors
. regress ltotexp $xlist $dlist if train==1, noheader vce(robust)
```

| ltotexp  | Coefficient | Robust    |       |       |                      |           |
|----------|-------------|-----------|-------|-------|----------------------|-----------|
|          |             | std. err. | t     | P> t  | [95% conf. interval] |           |
| income   | .0010653    | .0010664  | 1.00  | 0.318 | -.0010259            | .0031565  |
| educyr   | .0431495    | .0081645  | 5.29  | 0.000 | .027139              | .0591599  |
| age      | .0025177    | .0040582  | 0.62  | 0.535 | -.0054403            | .0104757  |
| famsze   | -.0635828   | .0285771  | -2.22 | 0.026 | -.1196218            | -.0075437 |
| totchr   | .3220218    | .0208646  | 15.43 | 0.000 | .2811068             | .3629368  |
| suppins  | .1547863    | .0523682  | 2.96  | 0.003 | .0520934             | .2574791  |
| female   | -.0643839   | .052321   | -1.23 | 0.219 | -.1669842            | .0382164  |
| white    | .1773761    | .1474569  | 1.20  | 0.229 | -.1117833            | .4665356  |
| hisp     | -.1031283   | .1030525  | -1.00 | 0.317 | -.3052118            | .0989552  |
| marry    | .1491644    | .0571793  | 2.61  | 0.009 | .0370372             | .2612917  |
| northe   | .2805731    | .0794206  | 3.53  | 0.000 | .1248312             | .436315   |
| mwest    | .3296948    | .0760097  | 4.34  | 0.000 | .1806417             | .478748   |
| south    | .1997139    | .0670176  | 2.98  | 0.003 | .068294              | .3311338  |
| msa      | .0677191    | .0572256  | 1.18  | 0.237 | -.044499             | .1799372  |
| phylim   | .2661041    | .0627222  | 4.24  | 0.000 | .1431074             | .3891008  |
| actlim   | .39576      | .0698797  | 5.66  | 0.000 | .2587277             | .5327924  |
| injury   | .1305469    | .0607895  | 2.15  | 0.032 | .0113402             | .2497537  |
| priolist | .3835745    | .077633   | 4.94  | 0.000 | .2313381             | .535811   |
| hvgg     | -.0965534   | .0505962  | -1.91 | 0.056 | -.1957713            | .0026646  |
| _cons    | 5.823748    | .3754025  | 15.51 | 0.000 | 5.087593             | 6.559903  |

```
. qui predict y_small
```

A second predictor fits an OLS regression on the full set of 188 interacted variables.

```
. * OLS with 188 potential regressors and 104 estimated
. qui regress ltotexp $rlist if train==1
. qui predict y_full
```

From suppressed output, the coefficients of 104 of the 188 variables are identified.

The third and fourth predictors we consider are penalized and postselection estimates of the coefficients from a lasso with tuning parameter  $\lambda$  determined by adaptive 10-fold cv. Note that for each of the 10 folds, this uses 90% of the 2,364 observations in the training sample for fitting and the remaining 10% for determining the best value of  $\lambda$  based on predictive ability.

```

. * LASSO with 188 potential regressors leads to 32 selected
. qui lasso linear ltotexp $rlist if train==1, selection(adaptive)
> rseed(10101) nolog
. lassoknots

```

| ID    | lambda   | No. of nonzero<br>coef. | CV mean<br>pred.<br>error | Variables (A)dded, (R)emoved,<br>or left (U)nchanged |
|-------|----------|-------------------------|---------------------------|------------------------------------------------------|
| 51    | 17.76327 | 1                       | 1.76889                   | A totchr                                             |
| 59    | 8.438993 | 2                       | 1.55272                   | A 0.actlim                                           |
| 66    | 4.400098 | 3                       | 1.473914                  | A 1.priolist#c.educyr                                |
| 71    | 2.76339  | 4                       | 1.430335                  | A 0.phylin#c.famsze                                  |
| 73    | 2.294215 | 6                       | 1.415289                  | A 1.marry#c.educyr<br>1.suppins#c.age                |
| 78    | 1.440834 | 7                       | 1.386716                  | A 0.hvgg#c.totchr                                    |
| 80    | 1.196205 | 9                       | 1.380092                  | A 1.mwest#c.totchr<br>1.injury#c.educyr              |
| 84    | .824498  | 10                      | 1.369338                  | A 1.mwest#c.famsze                                   |
| 85    | .7512519 | 11                      | 1.367485                  | A 0.female#c.totchr                                  |
| 87    | .6237025 | 12                      | 1.364392                  | A 0.priolist#c.totchr                                |
| 89    | .5178088 | 13                      | 1.361144                  | A 0.marry#c.totchr                                   |
| 90    | .4718081 | 14                      | 1.359738                  | A 1.northe#c.educyr                                  |
| 91    | .4298939 | 15                      | 1.35839                   | A 0.actlim#c.totchr                                  |
| 92    | .3917033 | 16                      | 1.356668                  | A 0.priolist#c.famsze                                |
| 95    | .2963092 | 17                      | 1.352067                  | A 1.south#c.educyr                                   |
| 96    | .2699859 | 18                      | 1.350489                  | A 0.white#c.famsze                                   |
| 99    | .2042345 | 20                      | 1.346719                  | A 1.female#c.income<br>1.phylin#c.educyr             |
| 100   | .1860908 | 21                      | 1.346044                  | A 0.actlim#c.famsze                                  |
| 101   | .169559  | 23                      | 1.345632                  | A 1.actlim#c.famsze<br>1.northe#c.totchr             |
| 103   | .1407709 | 25                      | 1.344879                  | A 0.south#c.famsze<br>0.injury#c.totchr              |
| 104   | .1282652 | 26                      | 1.344431                  | A 0.suppins#c.income                                 |
| 105   | .1168705 | 27                      | 1.344094                  | A 1.hvgg#c.educyr                                    |
| 106   | .106488  | 28                      | 1.343763                  | A 0.priolist                                         |
| 107   | .0970279 | 29                      | 1.343447                  | A 1.hisp#c.income                                    |
| 108   | .0884082 | 30                      | 1.343113                  | A 0.suppins#c.totchr                                 |
| 110   | .0733981 | 31                      | 1.342763                  | A 1.mwest#c.income                                   |
| 112   | .0609364 | 32                      | 1.341704                  | A 1.msa#c.educyr                                     |
| * 120 | .0289497 | 32                      | 1.339496                  | U                                                    |
| 121   | .0263779 | 33                      | 1.339525                  | A 1.hvgg#c.famsze                                    |
| 128   | .0137535 | 34                      | 1.340677                  | A 0.suppins#c.famsze                                 |
| 130   | .0114184 | 35                      | 1.341051                  | A 1.actlim#c.income                                  |
| 132   | .0094797 | 36                      | 1.341602                  | A 0.msa                                              |
| 135   | .0071711 | 38                      | 1.342449                  | A 1.hisp#c.age<br>1.south#c.totchr                   |
| 136   | .006534  | 37                      | 1.342685                  | R 1.msa#c.educyr                                     |
| 138   | .0054246 | 36                      | 1.343111                  | R 0.actlim#c.famsze                                  |
| 139   | .0049427 | 37                      | 1.343368                  | A 1.mwest#c.age                                      |
| 143   | .0034068 | 39                      | 1.34451                   | A 1.msa#c.educyr<br>1.northe#c.income                |
| 144   | .0031042 | 38                      | 1.344751                  | R totchr                                             |
| 145   | .0028284 | 39                      | 1.344983                  | A 0.actlim#c.famsze                                  |
| 147   | .0023482 | 40                      | 1.345372                  | A totchr                                             |
| 149   | .0019495 | 40                      | 1.345694                  | U                                                    |

\* lambda selected by cross-validation in final adaptive step.

```

. qui predict y_laspen // Use penalized coefficients
. qui predict y_laspost, postselection // Use post selection OLS coeffs

```

The adaptive lasso leads to 32 selected variables, and many are interactions such as `0.actlim#c.totchr`, the number of chronic conditions for individuals without an activity limitation. We then calculate two predictors. The first uses the penalized coefficients that are the lasso coefficient estimates. The second uses coefficients obtained by OLS regression on the 32 regressors selected by the lasso.

By comparison, the option `selection(cv)` leads to 42 selected variables, and the option `selection(bic)` leads to 17 selected variables.

As a clustered example, for illustrative purposes, we add the option `cluster(age)`. Then CV selects 46 variables, adaptive lasso selects 34 variables, and BIC selects 0 variables.

A fifth predictor regresses `ltotexp` on the first 5 principal components of the 19 underlying variables.

```
. * Principal components using the first 5 principal components of 19 variables
. qui pca $xlist $dlist if train==1
. qui predict pc*
. qui regress ltotexp pc1-pc5 if train==1
. qui predict y_pca
```

A sixth predictor is a neural network on the 19 underlying variables, computed using the community-contributed `brain` command ([Doherr 2018](#)). The option `hidden(10 10)`, for example, specifies 2 hidden layers with 10 hidden units in each layer. We fit a neural network with just 1 hidden layer and 10 units in that layer. Using program defaults, we obtain

```
. * Neural network with 19 variables and 2 hidden layers each with 10 units
. brain define, input($xlist $dlist) output(ltotexp) hidden(10)
Defined matrices:
 input[4,19]
 output[4,1]
 neuron[1,30]
 layer[1,3]
 brain[1,211]
. qui brain train if train==1, iter(500) eta(2)
. brain think y_neural
```

A seventh predictor is a random forest. The community-contributed `rforest` command ([Schonlau and Zou 2020](#)) estimates random forests for regression and classification. The `type(reg)` option specifies that the tree is for regression and not for classification; the `depth(10)` option limits the depth of the tree to be no more than 10; and the `lsize(5)` option sets the minimum number of observations per leaf to 5. Other options are set at their default values. This includes setting `numvars()`, the number of variables randomly selected for each tree, to the square root of the number of predictors, here 5 because it is the first integer to exceed  $\sqrt{19}$ .

```
. * Random forest with 19 variables
. qui rforest ltotexp $xlist $dlist if train==1,
> type(reg) iter(200) depth(10) lsize(5)
. qui predict y_ranfor
```

Finally, the community-contributed `boost` command ([Schonlau 2005](#)) accommodates boosting and bagging regression trees for linear, logistic, and Poisson regression. The command is a C++ plugin that must first be loaded into Stata. For details on the method and program, see [Schonlau \(2005\)](#).

We use the program defaults for boosted regression trees.

```
. * Boosting linear regression with 19 variables
. program boost_plugin, plugin using("C:\ado\personal\boost64.dll")
. qui boost ltotexp $xlist $dlist if train==1,
> distribution(normal) trainfraction(0.8) maxiter(100) predict(y_boost)
```

### 28.7.3 Comparison of predictors

We compare the prediction performance of the various predictors both in sample and out of sample.

```

. * Training MSE and test MSE for the various methods
. qui regress ltotexp
. qui predict y_noreg
. foreach var of varlist y_noreg y_small y_full y_laspen y_laspost y_pca
> y_neural y_ranfor y_boost {
 2. qui gen `var'errorsq = (`var' - ltotexp)^2
 3. qui sum `var'errorsq if train == 1
 4. scalar mse`var'train = r(mean)
 5. qui sum `var'errorsq if train == 0
 6. qui scalar mse`var'test = r(mean)
 7. display "Predictor: " "`var'" _col(21)
> " Train MSE = " %5.3f mse`var'train " Test MSE = " %5.3f mse`var'test
 8. }
Predictor: y_noreg Train MSE = 1.821 Test MSE = 2.063
Predictor: y_small Train MSE = 1.339 Test MSE = 1.492
Predictor: y_full Train MSE = 1.262 Test MSE = 1.509
Predictor: y_laspen Train MSE = 1.298 Test MSE = 1.491
Predictor: y_laspost Train MSE = 1.297 Test MSE = 1.493
Predictor: y_pca Train MSE = 1.397 Test MSE = 1.545
Predictor: y_neural Train MSE = 1.211 Test MSE = 1.808
Predictor: y_ranfor Train MSE = 1.047 Test MSE = 1.574
Predictor: y_boost Train MSE = 1.459 Test MSE = 1.664

```

The in-sample MSE is smallest for the most flexible models, notably, random forests and neural networks.

By comparison, the out-of-sample MSE in this example is lowest for the simpler models, notably, OLS with just 19 regressors and the lasso estimators.

The results here for neural networks, random forest, and boosting are based mainly on use of default options. More careful determination of tuning parameters for these methods could be expected to improve their predictive ability.

## 28.8 Machine learning for inference in partial linear model

Machine learning methods can lead to better prediction than the regression methods historically employed in applied microeconomics. But much microeconomic research is instead aimed at estimating the partial effect of a single variable, or estimation of one or a few parameters, after controlling for the effect of many other variables.

Machine learning methods have the potential to control for these other nuisance variables, but any consequent statistical inference on the parameters or partial effects of interest needs to control for the data mining of machine learning. As noted in section [28.3.4](#), we cannot directly perform inference on lasso and related estimators; we cannot naively use the lasso.

Instead, a semiparametric approach is taken where a model depends in part on parameters of interest and in part on “nuisance” functions of other variables. If estimation is based on a moment condition that satisfies an orthogonalization property defined in section [28.8.8](#), then inference on the parameters of interest may be possible.

The leading example to date is the estimation of parameters in a partial linear model, using lasso under the sparsity assumption that only a few of the many potential control variables are relevant. We focus on this case and the associated Stata commands introduced in version 16.

### 28.8.1 Partial effects in the partial linear model

We consider a setting where interest lies in measuring the partial effect on  $y$  of a change in variables  $\mathbf{d}$ , controlling for additional control variables.

A partial linear model for linear regression specifies

$$y = \mathbf{d}'\boldsymbol{\alpha} + g(\mathbf{x}_c) + u$$

where  $\mathbf{x}_c$  denotes selected control variables and  $g(\cdot)$  is a flexible function of  $\mathbf{x}_c$ . The parameter  $\alpha$  can be given a causal interpretation with the selection-on-observables-only assumption that  $E(u|\mathbf{d}, \mathbf{x}_c) = 0$ . The goal is to obtain a root- $N$  consistent and asymptotically normal estimator of the partial effect  $\alpha$

The partial linear model was introduced in section 27.6. There  $g(\cdot)$  was unspecified, and estimation was by semiparametric methods that required that there be few controls  $\mathbf{x}_c$  to avoid the curse of dimensionality.

Lasso methods due to [Belloni, Chernozhukov, and Hansen \(2014\)](#) and related articles instead allow for complexity in  $g(\cdot)$  by specifying  $g(\mathbf{x}_c) \simeq \mathbf{x}'\gamma + r$ , where  $\mathbf{x}$  consists of  $\mathbf{x}_c$  and flexible transformations of  $\mathbf{x}_c$  such as polynomials and interactions and  $r$  is an approximation error. The starting point is then that

$$y = \mathbf{d}'\alpha + \mathbf{x}'\gamma + r + u \quad (28.4)$$

The lasso is used in creative ways, detailed in this section, to select a subset of the few variables in the high-dimensional  $\mathbf{x}$  and to construct regressors in a consequent regression that yields an estimate of  $\alpha$  that is root- $N$  consistent and asymptotically normal, despite the data mining.

The estimate of  $\alpha$  is often called a causal estimate because if the model is well specified with a good set of controls, then an assumption of selection on observables only may be reasonable. But the method is also applicable without a causal interpretation.

A key assumption, called the sparsity assumption, is that only a small fraction of the  $\mathbf{x}$  variables are relevant. Let  $p$  be the number of potential control variables ( $\mathbf{x}$ ) and  $s$  be the number of variables in the true model, where  $p$  and  $s$  may grow with  $N$ , though at rates considerably less than  $N$ . The precise sparsity assumption varies with the model and estimation method. For the partialing-out estimator, for example, the sparsity assumption is that  $s/(\sqrt{N}/\ln p)$  is small. Additionally, the approximation error  $r$  is assumed to satisfy  $\sqrt{(1/N) \sum_{i=1}^N r_i^2} \leq c\sqrt{(s/N)}$  for some  $c > 0$ .

In this section, we present Stata commands that yield three different lasso-based estimators of  $\alpha$ .

### 28.8.2 Partial linear model application

We consider the same example as in section [28.7](#), with the change that we are interested in estimating the partial effect of having supplementary health insurance, so  $d$  in [\(28.4\)](#) is the single binary variable `suppins`.

```
. * Data for inference on suppins example: 5 continuous and 13 binary variables
. qui use mus203mepsmedexp, clear
. keep if ltotexp != .
(109 observations deleted)
. global xlist2 income educyr age famsze totchr
. global dlist2 female white hisp marry northe mwest south
> msa phylim actlim injury priolist hvgg
. global rlist2 c.($xlist2)##c.($xlist2) i.($dlist2) c.($xlist2)#i.($dlist2)
```

For later comparison, we fit by OLS models without and with all the interactions terms.

```
. * OLS on small model and full model
. qui regress ltotexp suppins $xlist2 $dlist2, vce(robust)
. estimates store OLSSMALL
. qui regress ltotexp suppins $rlist2, vce(robust)
. estimates store OLSFULL
. estimates table OLSSMALL OLSFULL, keep(suppins) b(%9.4f) se stats(N df_m r2)
```

| Variable | OLSSMALL         | OLSFULL          |
|----------|------------------|------------------|
| suppins  | 0.1706<br>0.0469 | 0.1868<br>0.0478 |
| N        | 2955             | 2955             |
| df_m     | 19.0000          | 99.0000          |
| r2       | 0.2682           | 0.3028           |

Legend: b/se

In this example, there is little change in the coefficient of `suppins` going from a model with 19 regressors to a model with 99 identified regressors, and the gain in  $R^2$  is modest. Supplementary insurance is associated with a

17%–19% increase in health spending, and the estimates are highly statistically significant.

### 28.8.3 Partialing-out estimator

The partialing-out method is obtained in several steps. First, for scalar regressor  $d$ , perform a lasso of  $d$  on  $\mathbf{x}$ , and obtain a residual  $u_d$  from OLS regression of  $d$  on the selected variables. Second, perform a lasso of  $y$  on  $\mathbf{x}$ , and obtain a residual  $u_y$  from OLS regression of  $y$  on the selected variables. Finally, obtain  $\hat{\alpha}$  by OLS regression of  $u_y$  on  $u_d$ .

More generally, if there are  $K$  key regressors of interest, then perform  $K$  separate lassos of each  $d_k$  on  $\mathbf{x}$  and  $K$  subsequent OLS regressions on the selected variables to obtain  $K$  separate residuals. The estimates  $\hat{\alpha}_1, \dots, \hat{\alpha}_K$  are obtained by OLS regression of  $u_y$  on all  $K$  residuals.

The partialing-out method is qualitatively similar to the Robinson differencing estimator presented in section [27.6](#), which instead used residuals from kernel regression. The method here requires a sparsity assumption that the number of nonzero coefficients in the true model is small relative to the sample size  $N$  and grows at a rate no more than  $\sqrt{N}$ . More precisely,  $s/(\sqrt{N}/\ln p)$  should be small, where  $p$  is the number of potential control variables and  $s$  is the number of variables in the true model.

The preceding algorithm for the partialing-out estimator does not extend to nonlinear models. Instead, the `semi` option provides a variation, termed “semipartialing-out”. First,  $u_d$  is obtained as for partialing out. Second, perform a lasso of  $y$  on  $d$  and  $\mathbf{x}$ , and obtain a residual  $u_y$  from OLS regression of  $y$  on  $d$  and the selected  $\mathbf{x}$  variables. Finally, obtain  $\hat{\alpha}$  by IV regression of  $u_y$  on  $d$  with “instruments”  $u_d$ . For further details, see in the *Methods and formulas* section of [LASSO] **poregress**.

### 28.8.4 The **poregress** command and related commands

Partialing-out lasso estimates can be obtained using the **poregress** command, which has syntax

```
poregress depvar varsofinterest [if] [in], controls([(alwaysvars)] othervars) [options]
```

The control variables are specified with the option `controls([(alwaysvars) ] othervars)`, where *alwaysvars* are controls to always be included and *othervars* are variables selected or deselected by the lasso. The penalty  $\lambda$  can be determined by a plugin formula (the default option `selection(plugin)`), by CV (option `selection(cv)`), by adaptive CV (option `selection(adaptive)`), or by BIC (option `selection(bic)`). For CV methods, the `rseed(#)` option should be used.

The `vce(cluster clustvar)` option provides cluster-robust standard errors. Additionally, if the `selection(cv)` or `selection(adaptive)` option is used, then the CV determines folds at the cluster level, and the lasso objective functions use the average of within-cluster averages.

The following related commands have similar syntax though use different algorithms: `xporegress` and `dsregress` for the linear model; `pologit`, `xpologit`, and `dslogit` for binary outcomes; `popoisson`, `xpopoisson`, and `dspoisson` for count data; and `poivregress` and `xpoivregress` for IV estimation in the linear model.

The `lassoinfo`, `lassoknots`, `cvplot`, `bicplot`, `lassocoef`, and `coefpath` commands provide additional information on the fitted models.

### 28.8.5 Plugin penalty parameter

The default is to use the plugin formula for  $\lambda$ , developed for use of the lasso in the current inference setting, rather than for prediction. The plugin value of  $\lambda$  leads to selection of fewer variables than the other methods. A good exposition of the plugin formula, and relevant references, is given in [Ahrens, Hansen, and Schaffer \(2018\)](#).

For the linear model with independent heteroskedastic errors, Stata sets the penalty parameter  $\lambda = c\sqrt{N}\Phi(1 - \{\gamma/(2p)\})$ , where  $c = 1.1$  and  $\gamma = 0.1/\ln\{\max(p, N)\}$ . Several studies find that these are good values for  $c$  and  $\gamma$ . The individual loadings for each regressor are

$\kappa_j = \sqrt{(1/N) \sum_{i=1}^N (x_{ij}\hat{\varepsilon}_i)^2}$ , where  $\mathbf{x}_j$  has been normalized to have mean 0 and variance 1 and  $\hat{\varepsilon}_i$  is a residual obtained by a sequence of first-stage lassos that is detailed in the Stata documentation. These settings are based on

linear regression with heteroskedastic errors. The option selection(plugin, homoskedastic) is used for homoskedastic errors.

If the option vce(cluster *clustvar*) is used, then the plugin value is the same as for heteroskedastic errors.

## 28.8.6 Partialing-out application

In the current application, the partialing-out lasso selects 21 variables and yields coefficient and standard error of `suppins` quite similar to the OLS results.

```
. * Partialing-out partial linear model using default plugin lambda
. poregress ltotexp suppins, controls($rlist2)

Estimating lasso for ltotexp using plugin
Estimating lasso for suppins using plugin

Partialing-out linear model
Number of obs = 2,955
Number of controls = 176
Number of selected controls = 21
Wald chi2(1) = 15.43
Prob > chi2 = 0.0001
```

| ltotexp | Robust      |           |      |       |                      |          |
|---------|-------------|-----------|------|-------|----------------------|----------|
|         | Coefficient | std. err. | z    | P> z  | [95% conf. interval] |          |
| suppins | .1839193    | .0468223  | 3.93 | 0.000 | .0921493             | .2756892 |

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

The `lassoinfo` command lists the number of variables selected by the separate lassos for the dependent variable and the single regressor of interest.

```
. * Lasso information
. lassoinfo
 Estimate: active
 Command: poregress
```

| Variable | Model  | Selection method | lambda  | No. of selected variables |
|----------|--------|------------------|---------|---------------------------|
| ltotexp  | linear | plugin           | .080387 | 12                        |
| suppins  | linear | plugin           | .080387 | 9                         |

In total, 21 variables were selected, and this exactly equals the sum of variables selected by the two distinct lassos ( $12 + 9 = 21$ ). So this example is unusual in that the two sets of selected variables are disjoint.

Because the lasso is applied to more than one variable, several of the postestimation commands need to name the variable of interest. For the lasso for the dependent variable, relevant commands are `lassoknots`, `for(ltotexp); lassocoef (., for(ltotexp)); bicplot, for(ltotexp); cvplot, for(ltotexp);` and `coefpath, for(ltotexp)`. The last two commands are applicable only if selection is by CV. For variable `suppins`, use instead `for(suppins)`.

The partialing-out estimated coefficient of `suppins` can also be obtained by manually performing each step of the algorithm. We have

```
. * Partialing out done manually
. qui lasso linear suppins $rlist2, selection(plugin)
. qui predict suppins_lasso, postselection
. qui generate u_suppins = suppins - suppins_lasso
. qui lasso linear ltotexp $rlist2, selection(plugin)
. qui predict ltotexp_lasso, postselection
. qui generate u_ltotexp = ltotexp - ltotexp_lasso
. regress u_ltotexp u_suppins, vce(robust) noconstant noheader
```

|           | Coefficient | Robust std. err. | t    | P> t  | [95% conf. interval] |
|-----------|-------------|------------------|------|-------|----------------------|
| u_ltotexp | .1839193    | .0468223         | 3.93 | 0.000 | .0921117 .2757268    |
| u_suppins |             |                  |      |       |                      |

The `postselection` option of the `lasso predict` postestimation command is used here because it predicts by OLS regression on the variables selected by the preceding `lasso` command.

## 28.8.7 Clustered errors application

As an example of clustered data, we refit the previous model with the `vce(cluster age)` option. Using the default plugin method to determine the penalty, we obtain

```
. * Cluster-robust partialing-out partial linear model using default plugin lambda
. poregress ltotexp suppins, controls($rlist2) vce(cluster age)

Estimating lasso for ltotexp using plugin
Estimating lasso for suppins using plugin

Partialing-out linear model Number of obs = 2,955
 Number of controls = 176
 Number of selected controls = 15
 Wald chi2(1) = 8.57
 Prob > chi2 = 0.0034
 (Std. err. adjusted for 26 clusters in age)
```

| ltotexp | Coefficient | Robust    |      |       |                      |
|---------|-------------|-----------|------|-------|----------------------|
|         |             | std. err. | z    | P> z  | [95% conf. interval] |
| suppins | .1686531    | .0576049  | 2.93 | 0.003 | .0557496 .2815566    |

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

Note: Lassos are performed accounting for clusters in age.

The estimated coefficient is then 0.1687 with cluster-robust standard error 0.0576.

```

. * Lasso information
. lassoinfo
 Estimate: active
 Command: poregress

```

| Variable | Model  | Selection method | lambda   | No. of selected variables |
|----------|--------|------------------|----------|---------------------------|
| ltotexp  | linear | plugin           | .8343593 | 8                         |
| suppins  | linear | plugin           | .8343593 | 7                         |

In total, 15 variables were selected, with 8 of these variables coinciding with the 21 selected in the preceding independence case; 60 variables were selected using CV; and 41 were selected using adaptive CV.

### 28.8.8 Orthogonalization

The partialing-out estimator of  $\alpha$  in the partial linear model is a two-step estimator. Unlike many two-step estimators, the asymptotic distribution of the second-step estimator of  $\alpha$  is not changed by the first-step estimation. This happens because the second-step estimation is based on a moment condition that satisfies a special orthogonalization condition.

Define  $\alpha$  as parameters of interest and  $\eta$  as nuisance parameters, and consider a two-step estimator that at the first step estimates  $\hat{\eta}$  and at the second step estimates  $\hat{\alpha}$  by solving

$$\sum_{i=1}^n \psi(\mathbf{w}_i, \alpha, \hat{\eta}) = \mathbf{0}$$

where  $\mathbf{w}_i$  denotes all variables. Then the asymptotic distribution of  $\hat{\alpha}$  is unaffected by first-step estimation of  $\eta$  if the function  $\psi(\cdot)$  satisfies the orthogonalization condition that

$$E\{\partial\psi(\mathbf{w}_i, \alpha, \eta)/\partial\eta\} = \mathbf{0} \quad (28.5)$$

See [Cameron and Trivedi \(2005\)](#), 201) or [Wooldridge \(2010\)](#), 410). The intuition is that if changing  $\eta$  does not in expectation change  $\psi(\cdot)$ , then noise in  $\hat{\eta}$  will not affect the distribution of  $\hat{\alpha}$ , at least asymptotically.

Now consider the partial linear model  $y = \alpha d + g(\mathbf{x}) + u$ , and define  $\eta_1 = E(d|\mathbf{x})$  and  $\eta_2 = E(y|\mathbf{x})$ . Estimates  $\hat{\eta}_1$  (and  $\hat{\eta}_2$ ) are obtained by OLS regression of  $d$  (and  $y$ ) on the components of  $\mathbf{x}$  selected by the lasso. The partialing-out estimator of  $\alpha$  is then obtained by OLS regression of  $(y - \hat{\eta}_2)$  on  $(d - \hat{\eta}_1)$ . This corresponds to solving the population moment condition  $E\{\psi(\mathbf{w}, \alpha, \eta_1, \eta_2)\} = 0$ , where

$\psi(\mathbf{w}, \alpha, \eta_1, \eta_2) = (d - \eta_1)\{(y - \eta_2) - \alpha(d - \eta_1)\}$ . To see this, recall that the OLS estimator for regression of  $y$  on scalar  $x$  solves

$\sum_i x_i u_i = \sum_i x_i(y_i - \beta x_i) = 0$  with corresponding population moment condition  $E\{x(y - \beta x)\} = 0$ .

The orthogonalization condition (28.5) is satisfied because

$$E\{\partial\psi(\mathbf{w}, \alpha, \eta_1, \eta_2)/\partial\eta_1\} = E\{-(y - \eta_2) + 2\alpha(d - \eta_1)\}$$

$$E\{\partial\psi(\mathbf{w}, \alpha, \eta_1, \eta_2)/\partial\eta_2\} = E\{-(d - \eta_1)\}$$

and these expectations equal zero using  $\eta_1 = E(d|\mathbf{x})$  and  $\eta_2 = E(y|\mathbf{x})$ .

The orthogonalization result is extraordinarily powerful. The two-step partialing-out approach can in principle be applied for any two-step estimator satisfying the orthogonalization condition, also called Neyman orthogonalization, and in principle, the first step may use machine learners other than the lasso.

### 28.8.9 Cross-fit partialing-out estimator

The cross-fit partialing-out estimator is an adaptation of the partialing-out method that reduces bias by separating the sample used for lasso predictions of  $y$  and the components of  $\mathbf{d}$  from the sample used for subsequent estimation of  $\alpha$ . The combination of an orthogonalized moment and cross-fitting is called double machine learning or debiased machine learning and leads to methods requiring weaker assumptions.

The sample is split into a larger part for the estimation of nuisance components and a smaller part for estimation of the parameters of interest. For simplicity, consider scalar  $d$  and  $\alpha$ . The larger sample is used for lasso of components of  $d$  on  $\mathbf{x}$  (and  $y$  on  $\mathbf{x}$ ) and for subsequent postselection OLS regression of  $d$  (and  $y$ ) on the selected variables that yields predictions  $\hat{d} = \mathbf{x}'\hat{\pi}_d$  (and  $\hat{y} = \mathbf{x}'\hat{\pi}_y$ ). The smaller sample is then used to compute residuals  $\tilde{u}_d = d - \mathbf{x}'\hat{\pi}_d$  and  $\tilde{u}_y = y - \mathbf{x}'\hat{\pi}_y$ , and subsequent OLS regression of  $\tilde{u}_y$  on  $\tilde{u}_d$  yields estimate  $\tilde{\alpha}$ .

This method reduces the complications of data mining by using one sample to obtain the coefficients for  $\hat{\pi}_d$  and  $\hat{\pi}_y$  and using a separate sample for estimating  $\tilde{\alpha}$ . Such sample splitting leads to a more relaxed sparsity assumption that the number of nonzero coefficients grows at a rate no more than  $N$  rather than  $\sqrt{N}$ . More precisely,  $s/(N/\ln p)$  should be small, where  $p$  is the number of potential control variables and  $s$  is the number of variables in the true model.

The preceding algorithm leads to efficiency loss because only part of the original sample is used at the second step to estimate  $\alpha$ . So  $K$ -fold cross fitting is used. For clarity, set  $K = 10$ . Then, for each  $k = 1, \dots, 10$ , we obtain estimates  $\hat{\pi}_{d,k}$  and  $\hat{\pi}_{y,k}$  using 90% of the data and apply these estimates to the remaining 10% of the sample to form residuals  $\tilde{u}_{d,k}$  and  $\tilde{u}_{y,k}$ . This yields 10 sets of residuals, each using 10% of the sample. The default method for the `xporegress` command stacks these residuals to form  $N$  residuals  $\tilde{u}_d$  and  $\tilde{u}_y$  for the full sample; subsequent OLS regression of  $\tilde{u}_y$  on  $\tilde{u}_d$  yields estimate  $\tilde{\alpha}$ . The option `technique(dm11)` leads to an alternative estimator  $\tilde{\alpha} = 1/10 \sum_{k=1}^{10} \tilde{\alpha}_k$ , where  $\tilde{\alpha}_k$  is obtained by regression of  $\tilde{u}_{y,k}$  on  $\tilde{u}_{d,k}$  in the  $k$ th fold.

The `xporegress` command for cross-fit partialing-out has syntax similar to the `poregress` command. We obtain

```

. * Cross-fit partialing-out (double/debiased) using default plugin
. xporegress ltotexp suppins, controls($rlist2) rseed(10101) nolog

Cross-fit partialing-out Number of obs = 2,955
linear model Number of controls = 176
 Number of selected controls = 31
 Number of folds in cross-fit = 10
 Number of resamples = 1
 Wald chi2(1) = 15.66
 Prob > chi2 = 0.0001

```

| ltotexp | Robust      |           |      |       |                      |          |
|---------|-------------|-----------|------|-------|----------------------|----------|
|         | Coefficient | std. err. | z    | P> z  | [95% conf. interval] |          |
| suppins | .1856171    | .0469096  | 3.96 | 0.000 | .093676              | .2775582 |

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type lassoinfo to see number of selected variables in each lasso.

Across the 10 folds, the number of selected variables ranged from 11 to 14 for ltotexp and from 7 to 11 for suppins.

```

. * Summarize the number of selected variables across the ten folds
. lassoinfo

 Estimate: active
 Command: xporegress

```

| Variable | Model  | Selection method | No. of selected variables |        |     |
|----------|--------|------------------|---------------------------|--------|-----|
|          |        |                  | min                       | median | max |
| ltotexp  | linear | plugin           | 11                        | 13     | 14  |
| suppins  | linear | plugin           | 7                         | 9      | 11  |

## 28.8.10 Double-selection estimator

The double-selection method performs a lasso of  $y$  on  $x$  and separate lassos of each component of  $d$  on  $x$ . Then  $\hat{\alpha}$  is the coefficient of  $d$  from OLS regression of  $y$  on  $d$  and the union of all components of  $x$  selected by the various lassos.

The method has the advantage of simplicity. It requires a sparsity assumption similar to that for the partialing-out estimator, and it is

asymptotically equivalent to the partialing-out estimator.

The `dsregress` command for double-selection estimation has syntax similar to the `xporegress` command. We obtain

```
. * Double-selection partial linear model using default plugin
. dsregress ltotexp suppins, controls($rlist2)
```

Estimating lasso for ltotexp using plugin

Estimating lasso for suppins using plugin

```
Double-selection linear model
Number of obs = 2,955
Number of controls = 176
Number of selected controls = 21
Wald chi2(1) = 15.30
Prob > chi2 = 0.0001
```

| ltotexp | Coefficient | Robust    |      |       |         | [95% conf. interval] |
|---------|-------------|-----------|------|-------|---------|----------------------|
|         |             | std. err. | z    | P> z  |         |                      |
| suppins | .1836224    | .0469429  | 3.91 | 0.000 | .091616 | .2756289             |

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

The coefficient of 0.1836 is very close to the partialing-out estimate of 0.1839.

## 28.9 Machine learning for inference in other models

[Belloni, Chernozhukov, and Wei \(2016\)](#) extend the methods for the partial linear model to a generalized partial linear model, in which case (28.4) becomes

$$E(y|\mathbf{d}, \mathbf{x}) = f(\mathbf{d}'\boldsymbol{\alpha} + \mathbf{x}'\boldsymbol{\gamma})$$

where the function  $f(\cdot)$  is specified. Now the partial effect of a change in  $\mathbf{d}$  is more complicated, being  $\boldsymbol{\alpha} \times f'(\mathbf{d}'\boldsymbol{\alpha} + \mathbf{x}'\boldsymbol{\gamma})$ . Partial effects in this single index model can be interpreted as in section 13.7.3. In the special case that  $f(\cdot) = \exp(\cdot)$ , the partial effects can be interpreted as semielasticities, and for a logit model with  $f(z) = e^z/(1 + e^z)$ , the partial effects can be interpreted in terms of the log-odds ratio; see section 10.5.

In this section, we illustrate these extensions. Additionally, we present extension of the partial linear model to the case where the regressors  $\mathbf{d}$  are endogenous.

### 28.9.1 Estimators for exponential conditional mean models

An exponential conditional mean variant of the partial linear model specifies  $E(y|\mathbf{d}, \mathbf{x}) = \exp(\mathbf{d}'\boldsymbol{\alpha} + \mathbf{x}'\boldsymbol{\gamma})$ . Because only  $\boldsymbol{\alpha}$  is consistently estimated, we cannot compute a marginal effect of a component of  $\mathbf{d}$  changing, but, given the exponential conditional mean specification, we can interpret each component of  $\boldsymbol{\alpha}$  as a semielasticity; see section 13.7.3.

Commands `popoisson`, `xpopoisson`, and `dspoisson` have syntax similar to command `poregress`. The method is detailed in *Methods and formulas* in [LASSO] **popoisson**.

These commands are applicable to any nonnegative dependent variable with exponential conditional mean and are not restricted to count data. So we could apply this method with dependent variable `totexp`, the level of health expenditures.

Nonetheless, we illustrate the method for a count, converting `totexp` to a count that takes values between 0 and 15. We compare standard Poisson regression estimates with the partialing-out estimate. We obtain

```
. * Exponential variant of partial linear model and partialing-out estimator
. generate ycount = floor(sqrt(totexp/500))
. summarize ycount
```

| Variable | Obs   | Mean     | Std. dev. | Min | Max |
|----------|-------|----------|-----------|-----|-----|
| ycount   | 2,955 | 2.633841 | 2.202957  | 0   | 15  |

```
. qui poisson ycount suppins $xlist2 $dlist2, vce(robust)
. estimates store PSMALL
. qui poisson ycount suppins $rlist2, vce(robust)
. estimates store PFULL
. qui popoisson ycount suppins, controls($rlist2) coef
. estimates store PPOLASSO
. estimates table PSMALL PFULL PPOLASSO, keep(suppins) b(%9.4f) se
> stats(N df_m k_controls_sel)
```

| Variable                  | PSMALL           | PFULL            | PPOLASSO         |
|---------------------------|------------------|------------------|------------------|
| suppins                   | 0.0602<br>0.0298 | 0.0645<br>0.0299 | 0.0666<br>0.0304 |
| N<br>df_m<br>k_controls^1 | 2955<br>19.0000  | 2955<br>99.0000  | 2955<br>22.0000  |

Legend: b/se

The partialing-out estimator with the option `coef` yields  $\hat{\alpha} = 0.0666$ , so private insurance is associated with 6.66% higher outcome  $y$ . The default is to instead report exponentiated coefficients such as  $\exp(0.0666) = 1.0689$ , in which case  $y$  is viewed as 1.0689 times higher.

## 28.9.2 Estimators for the logit model

A logistic variant of the partial linear model specifies

$E(y|\mathbf{d}, \mathbf{x}) = \Lambda(\mathbf{d}'\boldsymbol{\alpha} + \mathbf{x}'\boldsymbol{\gamma})$ , where  $\Lambda(z) = e^z/(1 + e^z)$ . Because only  $\boldsymbol{\alpha}$  is consistently estimated, we cannot compute a marginal effect of a component of  $\mathbf{d}$  changing, but, given the logistic specification, we can interpret each

component of  $\alpha$  as the impact on the log-odds ratio or, equivalently, each exponentiated component of  $\alpha$  as the impact on the odds ratio; see section 10.5.

Commands `pologit`, `xpologit`, and `dslogit` have syntax similar to command `poregress`. For details, see *Methods and formulas* in [LASSO] **pologit**.

These commands are applicable to any dependent variable that takes a value between 0 and 1. To illustrate the method for a binary variable, we convert `totexp` to a variable that takes value 1 if expenditures exceed \$4,000. The standard logit regression estimates for the odds ratio (option `or`) are compared with the partialing-out estimate. We obtain

```
. * Logit variant of partial linear model and partialing-out estimator
. generate dy = totexp > 4000
. tabulate dy
```

| dy    | Freq. | Percent | Cum.   |
|-------|-------|---------|--------|
| 0     | 1,661 | 56.21   | 56.21  |
| 1     | 1,294 | 43.79   | 100.00 |
| Total |       | 2,955   | 100.00 |

```
. qui logit dy suppins $xlist2 $dlist2, or vce(robust)
. estimates store LSMALL
. qui logit dy suppins $rlist2, or vce(robust)
. estimates store LFULL
. qui pologit dy suppins, controls($rlist2) coef
. estimates store LPOLASSO
. estimates table LSMALL LFULL LPOLASSO, keep(suppins) b(%9.4f) se
> stats(N df_m k_controls_sel)
```

| Variable                  | LSMALL           | LFULL            | LPOLASSO         |
|---------------------------|------------------|------------------|------------------|
| suppins                   | 0.2498<br>0.0898 | 0.2792<br>0.0936 | 0.2632<br>0.0892 |
| N<br>df_m<br>k_controls^1 | 2955<br>19.0000  | 2955<br>99.0000  | 2955<br>19.0000  |

Legend: b/se

The partialing-out estimator with the default option `or` yields estimate  $\exp(\hat{\alpha}) = 0.2632$ , so the odds ratio  $\Pr(y = 1|d, \mathbf{x}) / \Pr(y = 0|d, \mathbf{x})$  of having medical expenditures exceeding \$4,000 is 26.3% higher for those with supplementary insurance compared with those without supplementary insurance.

### 28.9.3 Partialing-out for IV estimation

For IV estimation, the most efficient estimator uses all available instruments according to standard asymptotic theory. But in practice this asymptotic theory can fail in typical sample sizes when there are too many instruments. This many-instruments problem can arise when a model is considerably overidentified, with many more instruments than endogenous regressors. But it can also arise in a just-identified model if there are many controls that lead to a low first-stage  $F$  statistic because the marginal contribution of the instruments becomes slight after inclusion of the many controls.

[Chernozhukov, Hansen, and Spindler \(2015\)](#) extend the partialing-out estimator to IV estimation of the linear model with selection of a subset of control variables from many controls or a subset of instruments from many instruments, or both.

The `poivregress` command provides partialing-out estimates of  $\alpha$  and  $\delta$  in the model

$$y = \mathbf{d}'\boldsymbol{\alpha} + \mathbf{w}'\boldsymbol{\delta} + \mathbf{x}'\boldsymbol{\gamma} + v$$

where  $\mathbf{d}$  are endogenous variables,  $\mathbf{w}$  are exogenous variables to always be included, and  $\mathbf{x}$  are exogenous control variables that may potentially be included. Additionally, there are instruments  $\mathbf{z}$  with  $\text{dim}[\mathbf{z}] \geq \text{dim}[\mathbf{d}]$ .

The `poivregress` command has syntax

```
poivregress depvar [exovars] (endovars=instrumvars) [if] [in],
controls([(alwaysvars)] othervars) [options]
```

For simplicity, consider the case of scalar endogenous regressor  $d$  and  $\delta = 0$ . The partialing-out algorithm is the following.

1. Calculate a partialled-out independent variable as the residual  $\widehat{u}_{yi}$  from OLS regression of  $y$  on  $\tilde{\mathbf{x}}_y$ , where  $\tilde{\mathbf{x}}_y$  denotes the selected variables from a lasso of  $y$  on  $\mathbf{x}$ .
2. Calculate a scalar instrument  $\check{u}_{di}$  as follows. Perform a lasso of  $d$  on  $\mathbf{x}$  and  $\mathbf{z}$ , and denote the selected variables as, respectively,  $\tilde{\mathbf{x}}_d$  and  $\tilde{\mathbf{z}}_d$ . Then, obtain a prediction  $\widehat{d}$  from OLS regression of  $d$  on  $\tilde{\mathbf{x}}_d$  and  $\tilde{\mathbf{z}}_d$ . Then, calculate the residual  $\check{u}_{di}$  and the coefficients  $\check{\beta}$  from OLS regression of  $\widehat{d}$  on  $\check{\mathbf{x}}_{\widehat{d}}$ , where  $\check{\mathbf{x}}_{\widehat{d}}$  denotes the selected variables from a lasso of  $\widehat{d}$  on  $\mathbf{x}$ .
3. Calculate a partialled out endogenous regressor  $\widehat{u}_{di} = d_i - \check{\mathbf{x}}'_{\widehat{d}_i} \check{\beta}$ .
4. Compute  $\widehat{\alpha}$  by IV regression of  $\widehat{u}_{yi}$  on  $\widehat{u}_{di}$  with  $\check{u}_{di}$  as the instrument.

As an example, we consider a variant of the analysis of [Belloni et al. \(2012\)](#) based on [Acemoglu, Johnson, and Robinson \(2001\)](#). In this example, the model is just identified, but there are only 64 observations and 24 potential controls that could lead to a weak instrument problem.

The goal is to use a cross-sectional sample of countries to measure the causal effect on per capita income (`logpgp95`) of protection against expropriation risk (`avexpr`). The mortality rate of early settlers (`logem4`) is used as an instrument for `avexpr`. The global macro `x2list` includes 24 possible control variables that include measures of country latitude, temperature, humidity, soil types, and natural resources.

```

. * Read in Acemoglu-Johnson-Robinson data and define globals
. qui use mus228ajr, clear
. global xlist lat_abst edes1975 avelf temp* humid* steplow deslow
> stepmid desmid drystep drywint goldm iron silv zinc oilres landlock
. describe logpgp95 avexpr logem4
Variable Storage Display Value
 name type format label Variable label

logpgp95 float %9.0g
avexpr float %9.0g
logem4 float %9.0g
. summarize logpgp95 avexpr logem4, sep(0)
 Variable Obs Mean Std. dev. Min Max
-----+
logpgp95 64 8.062237 1.043359 6.109248 10.21574
avexpr 64 6.515625 1.468647 3.5 10
logem4 64 4.657031 1.257984 2.145931 7.986165

```

We use command `poivregress` with plugin bandwidth for the homoskedastic case.

```

. * Partialing-out IV using plugin for lambda
. poivregress logpgp95 (avexpr=logem4), controls($xlist) selection(plugin, hom)
Estimating lasso for logpgp95 using plugin
Estimating lasso for avexpr using plugin
Estimating lasso for pred(avexpr) using plugin
Partialing-out IV linear model Number of obs = 64
 Number of controls = 24
 Number of instruments = 1
 Number of selected controls = 5
 Number of selected instruments = 1
 Wald chi2(1) = 8.74
 Prob > chi2 = 0.0031
-----+
 Robust
 logpgp95 Coefficient std. err. z P>|z| [95% conf. interval]
-----+
 avexpr .8798503 .2976286 2.96 0.003 .296509 1.463192

```

Endogenous: avexpr

Note: Chi-squared test is a Wald test of the coefficients of the variables of interest jointly equal to zero. Lassos select controls for model estimation. Type `lassoinfo` to see number of selected variables in each lasso.

Only six controls are selected. From `lassocoef(., for(avexpr))`, the first lasso selected `logem4`, `edes1975` and `zinc`; from

`lassocoef(., for(logpgp95))`, the second lasso selected `edes1975` and `avelf`; and from `lassocoef(., for(pred(avexpr))`), the third lasso selected `edes1975`, `avelf`, `temp2`, `iron`, and `zinc`.

From output not given, when all 24 controls are used, the regular IV estimate of `avexpr` is 0.713 with standard error 0.147. The reason for the smaller standard error of regular IV is that regular IV used an additional 19 controls that greatly improved model fit. At the same time, reducing the number of controls in the first-stage estimation led to more precise estimation of the first-stage coefficient of the instrument `logem4`.

The following code manually implements the partialing-out IV estimator for this example.

```
. * poivregress estimator in just-identified model obtained manually
. gen y = logpgp95
. gen d = avexpr
. global zlist logem4
. qui lasso linear y $xlist, selection(plugin, hom) // Lasso of y on x
. qui predict yhat, postselection
. generate yresid = y - yhat // Generate y residual
. qui lasso linear d $xlist $zlist, selection(plugin, hom) // Lasso d on x,z
. qui predict dhat, postselection // Generate dhat
. qui lasso linear dhat $xlist, selection(plugin, hom) // Lasso dhat on x
. predict dhat_hat, postselection
(option xb assumed; linear prediction with postselection coefficients)
. generate dhatresid = dhat - dhat_hat // Generate dhat residual
. generate dresid = d - dhat_hat // Generate d "residual"
. ivregress 2sls yresid (dresid = dhatresid), noconstant vce(robust)

Instrumental variables 2SLS regression
 Number of obs = 64
 Wald chi2(1) = .
 Prob > chi2 = .
 R-squared = .
 Root MSE = .81396
```

|        | Robust      |           |      |       |                      |          |
|--------|-------------|-----------|------|-------|----------------------|----------|
|        | Coefficient | std. err. | z    | P> z  | [95% conf. interval] |          |
| yresid |             |           |      |       |                      |          |
| dresid | .8798503    | .2952943  | 2.98 | 0.003 | .3010841             | 1.458617 |

Instrumented: `dresid`  
Instruments: `dhatresid`

The estimate equals that from `poivregress`, while the slight difference in the standard error is due to different degrees-of-freedom correction.

Care is needed in using the `poivregress` command because it is possible that the lasso of  $d$  on  $x$  and  $z$  may lead to too few instruments being selected, in which case the model becomes unidentified. Indeed, this happened in the current example when the default heteroskedastic variant of the plugin value of `lambda` was used, because then the single instrument `logem4` in this just-identified example was not selected.

Such selection of too few instruments is even more likely to occur with the cross-fitting `xpoivregress` command because the variable selection then occurs  $K$  times. The remedy is to use a larger value of the lasso penalty  $\lambda$ .

#### 28.9.4 Further discussion

The methods illustrated have been restricted to the partial linear model and generalized partial linear model using lasso under the assumption of sparsity. These examples can be extended to the use of other machine learners and to application in other models.

[Farrell \(2015\)](#) applies the lasso to the doubly robust augmented inverse-probability weighting estimator of average treatment effect (ATE) for a binary treatment presented in section [24.6.5](#). The `telasso` command, introduced in Stata 17, implements this estimator. The command syntax, similar to that for `teffects` commands, is

```
telasso ipwra (ovar omvarlist [, omodel om_options])
 (tvar tmvarlist [, tmodel tm_options]) [if] [in] [weight] [, stat options]
```

where `om_options` and `tm_options` include options for the lasso and determination of the lasso penalty parameter in, respectively, the outcome model and the treatment model. The outcome model can be linear, logit, probit, or Poisson; the binary treatment model can be logit or probit; and the command can compute ATE, ATE on the treated, and potential-outcome means. For details on the implementation of the lasso, see especially the `lasso` command in section 28.4.1. The `telasso` command option `vce(cluster clustvar)` provides cluster-robust standard errors. Note that in

this case the lasso is one that gives equal weight to each cluster rather than to each observation.

[Farrell, Liang, and Misra \(2021\)](#) establish theory suitable for use of deep nets for causal inference and provide an application using the augmented inverse-probability weighting TES estimator.

Many of these methods use orthogonalized moment conditions (see section [28.8.8](#)) and cross-fitting (see section [28.8.9](#)). [Chernozhukov et al. \(2018\)](#) provide an excellent overview, theory, and applications that use a variety of machine learners (lasso, regression tree, random forest, boosting, and neural network) to estimate ATE and local ATE for a binary treatment with heterogeneous effects and for IV estimation in a partial linear model. The machine learner needs to approximate well the nuisance part of the model. Appropriate assumptions to ensure this will vary with the setting and will not necessarily require a sparsity assumption.

This is an exceptionally active area of current econometric research, and we anticipate an explosion of new methods that will be implementable in Stata using one's own coding as community-contributed Stata programs and, ultimately in some cases, as official Stata programs.

In a separate important strand of research that is not covered here, [Wager and Athey \(2018\)](#) use random forests to estimate the ATE for a binary treatment for subgroups of the population and to identify groups with the greatest TE. They provide nonstandard asymptotic results that yield pointwise confidence intervals. Their method includes the use of “honest trees”, qualitatively similar to cross-fitting. [Wager and Athey \(2019\)](#) provide an empirical example.

Machine learning methods for causal inference rely heavily on asymptotic theory, and investigation of finite-sample behavior is quite recent. For example, [Wüthrich and Zhu \(Forthcoming\)](#) find poor finite-sample confidence interval coverage for the partial linear model estimated using the double-selection lasso. They instead suggest as an alternative direct OLS estimation with the many controls included as regressors, and subsequent inference based on recently developed methods for inference when there are many covariates.

## 28.10 Additional resources

Machine learning methods have only recently been used in microeconometrics. References that are written from a statistics perspective include a master's level text by [James et al. \(2021\)](#) and more advanced texts by [Hastie, Tibshirani, and Friedman \(2009\)](#) and [Efron and Hastie \(2016\)](#). [Varian \(2014\)](#) provides an early summary for economists of machine learning approaches and some of the software packages developed to handle massive datasets. [Mullainathan and Spiess \(2017\)](#) provide a detailed application of machine learning methods for prediction in economics and cite many applications. [Athey and Imbens \(2019\)](#) provide a more recent overview. [Hansen \(2022\)](#), chap. 29) covers both machine learning methods and their use for causal inference.

Initial research on use of machine learning for statistical inference has used the lasso; see [Belloni, Chernozhukov, and Hansen \(2014\)](#) for an accessible summary and illustration. [Drukker \(2020\)](#) provides a more recent account. Stata 16 introduced commands for lasso and elastic net for both prediction and statistical inference that are detailed in [LASSO] *Stata Lasso Reference Manual*.

The `lassopack` package (Ahrens, Hansen, and Schaffer [2019](#)) overlaps considerably with the Stata lasso commands for prediction. The accompanying article is well worth reading because it provides details on the background theory. The community-contributed `pdslasso` and `ivlasso` commands ([Ahrens, Hansen, and Schaffer 2018](#)) overlap considerably with the Stata commands for inference in the partial linear model and include some additional features.

Other machine learning methods may be used, both for prediction and for statistical inference. The important innovation of double or debiased machine learning with orthogonalized moment conditions and cross fitting is presented in [Chernozhukov et al. \(2018\)](#); see also [Chernozhukov, Newey, and Singh \(2022\)](#). [Wager and Athey \(2018\)](#) and related articles use random forests to estimate heterogeneous treatment effects.

The community-contributed `pylearn` package ([Droste 2020](#)) provides functions that implement popular python functions for regression trees, random forests, neural networks, adaptive boosting, and gradient boosting.

## 28.11 Exercises

1. Suppose we have a sample of 8 observations for which  $y$  takes values 1, 2, 3, 4, 5, 6, 7, and 8. We wish to predict  $y$  using the sample mean. Compute MSE using all the data. Now, suppose we choose as training dataset the first, third, fifth, and seventh observations, and the remaining four observations are the test data. Compute the test MSE. Now, suppose we use four-fold CV where the first fold has the first and fifth observations, the second fold the second and sixth, the third fold the third and seventh, and the fourth fold the fourth and eighth. Compute MSE for each fold. Hence, compute CV4 and the standard error of CV4.
2. Repeat the analysis of section [28.4.6](#) with regressors  $x_1$ ,  $x_2$ , and  $x_3$  augmented by their products and cross products, again using `rseed(10101)`. Comment on the differences between the various estimates.
3. Generate a sample of 10,000 observations using the following code.

```
set obs 10000
set seed 10101
matrix MU = (0,0,0)
scalar rho = 0.95
matrix SIGMA = (1,rho,rho \ rho,1,rho \ rho,rho,1)
drawnorm x1 x2 x3, means(MU) cov(SIGMA)
scalar rho = 0.2
matrix SIGMA = (1,rho,rho \ rho,1,rho \ rho,rho,1)
drawnorm x4 x5 x6, means(MU) cov(SIGMA)
generate y = 1 + 2*x1 + 3*x2 + 2*x1*x2 + 2*x4 + 3*x5 + 2*x4*x5 + rnormal(0,10)
```

The potential regressors are  $x_1$ – $x_6$  and their products and cross products. Perform adaptive lasso with the option `rseed(101010)` on the full sample, on the first 1,000 observations, and on the first 100 observations. Comment on the ability of lasso to detect the true model as the sample sizes changes. (This comparison is easier following the example in section [28.4.6](#).) Similarly, perform OLS of  $y$  on all potential regressors. Suppose we select only those regressors that are statistically significant at 5%. Comment on the ability of OLS to detect the true model as the sample sizes changes. (This comparison is simpler using the `star` option of `estimates table`.)

4. Perform an analysis with training and test samples qualitatively similar to those in section [28.7](#) using the same generated data as in section [28.2](#). Specifically, split the data into a training sample of 30 observations and a test sample of 10 observations, using command

```
splitsample y, generate(train) split(1 3) values(0 1) rseed(10101)
```

Use OLS and lasso with five-fold CV to obtain predictions. In the case of lasso, obtain predictions from both penalized coefficients and postselection coefficients. Which method predicts best in the training sample? Which method predicts best in the holdout sample?

5. Use the same data as in section [28.7](#), but use only the continuous regressors, so the regressor list is `c.($xlist)##c.($xlist)`. Give the same `splitsample` command as in question 4 above. Using the training dataset, perform 10-fold CV lasso, adaptive lasso, ridge, and elasticnet regression with the option `rseed(101010)`. Compare the coefficients selected and their penalized values, training sample fit, and test sample fit across the estimates.
6. Repeat the previous question but using an exponential conditional mean model rather than a log-linear model. The dependent variable is `totexp`, the level of expenditure; the sample should now include observations with `totexp = 0`; the `lasso poisson` and `elasticnet poisson` commands are used; and selected variables are compared with Poisson estimates with heteroskedastic-robust standard errors that are statistically significant at 5%.
7. Fit a partial linear model using the same generated sample as that in question 3. The single regressor of interest is `x1`, and the potential controls are all other variables and interactions, including interactions with `x1`. The list of potential controls can be set up using commands

```
global xlist2 x2 x3 x4 x5 x6
global x1interact c.x1c.($xlist2)
global rlist2 $x1interact c.($xlist2)c.($xlist2)
```

Use command `poregress` with default options on the full sample, on the first 1,000 observations, and on the first 100 observations. Compare the estimated coefficient of `x1` with the DGP value as the sample size changes. Comment on the controls chosen for `y` and `x1` as the sample size changes. Fit by OLS the model with all potential regressors included, and compare the coefficient of `x1` (and its heteroskedastic-robust standard error) with the `poregress` estimates. Fit using command `xporegress` with default options, and compare with the `poregress` estimates.

8. Use the same data as in section [28.8](#), and regress `ltotexp` on `suppins` and on controls that are only the continuous regressors, so the controls regressor list is `c.($xlist2)##c.($xlist2)`. Regress `ltotexp` on `suppins` and potential controls using commands `poregress`, `xporegress`, and `dsregress` with

default options and by OLS. Compare the fitted coefficients and their heteroskedastic-robust standard errors across these methods.



# **Chapter 29**

## **Bayesian methods: Basics**

## 29.1 Introduction

Bayesian methods combine prior information on parameters with a likelihood function for the data-generating process. The prior information expresses the investigator's uncertainty about the parameters, and the likelihood is a parametric expression of the conditional distribution of the data. The Bayesian approach provides a method of combining the two and is an alternative statistical inference framework to classical statistics and an alternative computational method for model fitting. The objective in Bayesian analysis is to obtain an estimate of the posterior distribution of the model parameters.

Until recently, Bayesian methods were infrequently used because of 1) analytical intractability in all but the simplest models; 2) lack of good prior information on parameters in many settings such as regression with many regressors and hence many parameters; and 3) resistance from classical statisticians to the Bayesian approach to inference.

The 1980's development of Markov chain Monte Carlo (MCMC) methods for estimating the posterior distribution using modern computing power has greatly expanded the range of models that can be fit using Bayesian methods, so the first concern is much less of an impediment.

The second concern, if relevant, can be mitigated by specifying prior information on data that is noninformative, meaning that it has little impact on the resulting estimates. And even if the specified prior information on parameters  $\beta$  is very strong, in large samples, the likelihood dominates and the prior has little effect.

The third concern, if relevant, can be overcome by using noninformative prior information and interpreting the posterior mode, defined in the next section, as the maximum likelihood estimator (MLE).

The `bayes` prefix makes Bayesian estimation as straightforward as using standard regression commands such as `regress` and `probit`. The `bayesmh` command allows estimation of a wider range of models, though requires

specification of an appropriate density for the data and a prior distribution for the parameters.

Regardless of which command is used, great care is needed in applying Bayesian methods. First, there is no guarantee that the MCMC iterative procedure has converged. There are diagnostics but there is no formal test; see section [29.4.4](#). Second, it is easy to specify an “uninformative” prior on a parameter that is in fact quite informative and can greatly influence the posterior distribution of all parameters; see section [29.6.4](#). Third, one can specify models that are nonidentified or weakly identified yet get seemingly sensible results; see section [29.6.5](#).

In this chapter, we focus on illustrating the simplest Bayesian methods for the linear regression model with normal homoskedastic errors and for the probit binary outcome model. The subsequent chapter [30](#) presents Bayesian MCMC methods in further detail and their use as one method for multiple imputation of missing data.

## 29.2 Bayesian introductory example

As an introduction, before any explanation of the methods, we estimate a linear regression using the simplest Bayesian command, the `bayes: regress` command. The samples used in this chapter are small to speed up computations and because then the prior can have greater influence on estimates.

### 29.2.1 The `bayes` prefix

The `bayes` prefix has syntax

```
bayes [, bayesopts]: estimation_command [, estopts]
```

where the *estimation\_command* covers over 60 fully parametric model commands and the many *bayesopts* include those specifying the priors.

The models covered include the leading models for linear regression (for example, `regress`, `tobit`); binary outcomes (`logit`, `probit`); multinomial outcomes (`ologit`, `mlogit`, `clogit`); counts (`poisson`, `nbreg`); generalized linear models (`glm`); duration models (`streg`); sample-selection models (`heckman`) and multilevel models (`mixed`, `melogit`, `meglm`); and random-effects (RE) models (`xtlogit`, `xtpoisson`, ...).

We illustrate the `bayes` prefix using the `bayes: regress` command. Later sections use the more flexible `bayesmh` command, though most examples could have been implemented using the simpler `bayes` prefix.

### 29.2.2 Bayesian estimates using default priors

The data are based on a random sample of 100 observations for men and women aged 25 to 65 years who were full-time workers in 2010, extracted from the American Community Survey.

```

. * Read in earnings - schooling data
. qui use mus229acs
. describe earnings lnearnings age education

```

| Variable<br>name | Storage<br>type | Display<br>format | Value<br>label | Variable label                                |
|------------------|-----------------|-------------------|----------------|-----------------------------------------------|
| earnings         | float           | %9.0g             |                | Annual earnings in \$                         |
| lnearnings       | float           | %9.0g             |                | Natural logarithm of earnings                 |
| age              | int             | %36.0g            |                | Age in years                                  |
| education        | float           | %9.0g             |                | Educational attainment: years of<br>schooling |

```

. keep if _n <= 100
(772 observations deleted)
. summarize earnings lnearnings age education

```

| Variable   | Obs | Mean     | Std. dev. | Min      | Max      |
|------------|-----|----------|-----------|----------|----------|
| earnings   | 100 | 60244    | 46513.19  | 4000     | 318000   |
| lnearnings | 100 | 10.76058 | .7273709  | 8.294049 | 12.66981 |
| age        | 100 | 43.33    | 10.9342   | 25       | 65       |
| education  | 100 | 13.69    | 3.158106  | 0        | 20       |

We obtain Bayesian estimates using the `bayes: regress` command with default options. This is simply the usual command one would use for linear regression, prefixed by `bayes`.

Because estimates are obtained using simulation methods, explained below, we set the seed to ensure reproducibility of results. We use the `rseed()` option of the `bayes` prefix or `bayesmh` command. In many cases, the `set seed` command could equivalently be used, but this is not the case, for example, if multiple chains are used because then the same seed would be erroneously set for each chain.

```

. * Bayesian linear regression with uninformative prior
. bayes, rseed(10101): regress lnearnings education age
Burn-in ...
Simulation ...
Model summary

```

---

Likelihood:

lnearnings ~ regress(xb\_lnearnings,{sigma2})

Priors:

{lnearnings:education age \_cons} ~ normal(0,10000) (1)  
{sigma2} ~ igamma(.01,.01)

---

(1) Parameters are elements of the linear form xb\_lnearnings.

|                                          |                  |     |        |
|------------------------------------------|------------------|-----|--------|
| Bayesian linear regression               | MCMC iterations  | =   | 12,500 |
| Random-walk Metropolis-Hastings sampling | Burn-in          | =   | 2,500  |
|                                          | MCMC sample size | =   | 10,000 |
|                                          | Number of obs    | =   | 100    |
|                                          | Acceptance rate  | =   | .3071  |
|                                          | Efficiency: min  | =   | .07066 |
|                                          |                  | avg | .09299 |
|                                          |                  | max | .1512  |

Log marginal-likelihood = -133.37046

|            | Mean     | Std. dev. | MCSE    | Median   | Equal-tailed<br>[95% cred. interval] |          |
|------------|----------|-----------|---------|----------|--------------------------------------|----------|
|            |          |           |         |          |                                      |          |
| lnearnings |          |           |         |          |                                      |          |
| education  | .0871874 | .0217776  | .000819 | .0868041 | .0471493                             | .1312628 |
| age        | .008496  | .0062873  | .000231 | .0089316 | -.0037933                            | .0208249 |
| _cons      | 9.198406 | .4482471  | .016292 | 9.196124 | 8.319206                             | 10.09851 |
| sigma2     | .4774248 | .0711248  | .001829 | .4702676 | .3587335                             | .6308758 |

Note: Default priors are used for model parameters.

Note: Adaptation tolerance is not met in at least one of the blocks.

Bayesian methods combine prior information on model parameters with the information on these parameters obtained from the likelihood function of the data to obtain the posterior distribution of the parameters.

The first set of output details the likelihood and priors. The second set of output gives details on the computational method used and is analogous to an iteration log. The MCMC iterative procedure yields 10,000 draws, called posterior draws, of each of the parameters. The final set of output summarizes the distribution of these draws.

We consider each set of output in detail; more detailed explanation is given in section [29.4](#).

The first set of output states that the likelihood function is the linear regression model with independent and identically distributed (i.i.d.) normal errors, the priors for the regression intercept and slope parameters are  $N(0, 100^2)$ , and the prior for the error variance is an inverse-gamma distribution with shape parameter 0.01 and scale parameter 0.01. The intent of these defaults is that the variance of these priors is so large that the priors are uninformative, meaning that they do not have much impact on the results. This need not necessarily be the case. For example, if the intercept is 10,000, then a  $N(0, 100^2)$  prior may have an impact. Going the other way, we may have strong prior beliefs, in which case we should provide priors that reflect this information. Options of the `bayes` prefix enable one to change the priors from the defaults.

The second set of output states that 12,500 draws of the parameters were made, the first 2,500 were discarded (the burn-in), and the remaining 10,000 were kept. These 10,000 draws are not independent of each other. The average sampling efficiency statistic of 0.09299 means that on average for the 4 parameters, the 10,000 correlated draws contain as much information as if  $10000 \times 0.09299 = 930$  independent draws had been made.

The final set of output summarizes the results. As expected, earnings increase with education and age. Consider the results for  $\beta_{ed}$ , the coefficient of `education`. The 10,000 posterior draws have mean 0.0872 and standard deviation 0.0218, and the interval between the 2.5 percentile and 97.5 percentile of the 10,000 draws is [0.0471, 0.1313]. A purely Bayesian interpretation of the results is that the posterior distribution of the values that the parameter  $\beta_{ed}$  may take has mean 0.0872, standard deviation 0.0218, and a 95% probability that  $\beta_{ed}$  lies in the interval [0.0471, 0.1313]. The `MCSE` column provides the Monte Carlo standard error, defined in section [29.4.4](#), that should be small relative to the corresponding standard deviation. That is the case here.

The Bayesian and maximum likelihood (ML) approaches are compared in some detail in section [29.3.6](#). In this example, the MLE for the regression parameters is just the ordinary least-squares (OLS) estimator. We obtain

```
. * ML linear regression (same as OLS with i.i.d. errors)
. regress lnearnings education age, noheader
```

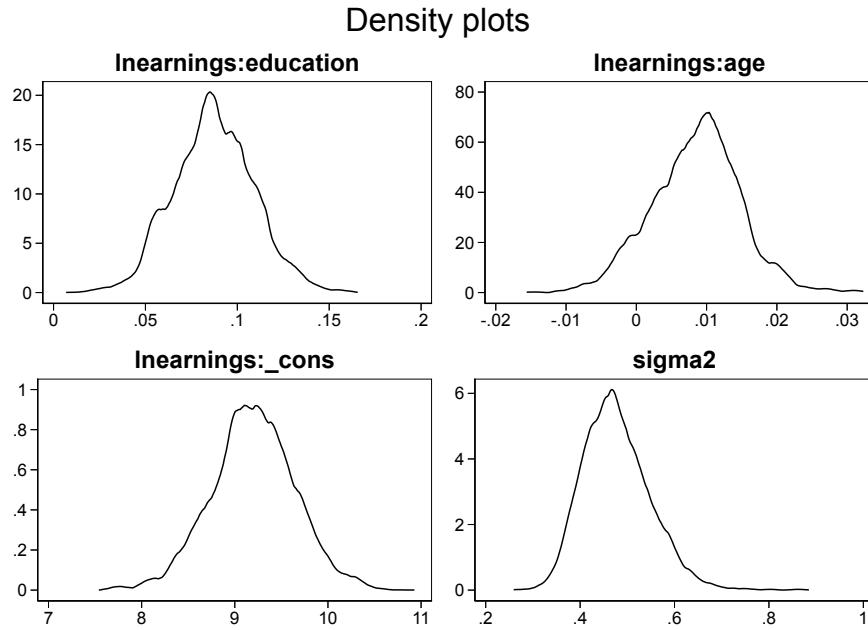
| lnearnings | Coefficient | Std. err. | t     | P> t  | [95% conf. interval] |
|------------|-------------|-----------|-------|-------|----------------------|
| education  | .0852959    | .0221804  | 3.85  | 0.000 | .0412739 .1293178    |
| age        | .0079952    | .0064063  | 1.25  | 0.215 | -.0047195 .02071     |
| _cons      | 9.246449    | .4546021  | 20.34 | 0.000 | 8.34419 10.14871     |

The ML estimate for `education` is 0.0853 with standard error 0.0222 and a 95% confidence interval for  $\beta_{\text{ed}}$  of [0.0413, 0.1293]. In this example, with an uninformative prior, the ML coefficients, standard errors, and 95% confidence intervals are similar to the Bayesian posterior means, posterior standard deviations, and 95% credible regions.

A key feature of Bayesian MCMC methods is that they yield an estimate of the distribution of parameters. To highlight this, we use the `bayesgraph kdensity` command to plot the densities of the 10,000 draws for each of the 4 model parameters.

```
. * Plot of the density of the 10,000 draws for each parameter
. quietly bayes, rseed(10101): regress lnearnings education age
. bayesgraph kdensity _all, combine
```

Figure 29.1 gives the density plots. The benefit of having draws of the parameters is that one can then easily do inference on any transformation of the parameters. For monotonic transformations  $g(\beta)$ , such as  $g(\beta) = \exp(\beta)$  or  $g(\beta) = \exp(\beta x)$ , a 95% credible interval is the interval between the 2.5 percentile and 97.5 percentile of the 10,000 values of  $g(\beta)$ . There is no need to use the delta method.



**Figure 29.1.** Posterior densities of the three regression parameters and the error variance parameter

It is important to look at various diagnostics to ensure that the computational method has led to sensible results. In particular, the MCMC iterative procedure needs to have converged. Various checks and potential pitfalls are presented in this chapter.

## 29.3 Bayesian methods overview

In this section, and more generally in this chapter, we provide a very brief introduction to Bayesian methods. Bayesian analysis is presented in many complete books, including the econometrics text of [Koop \(2003\)](#) and the statistics text of [Gelman et al. \(2013\)](#). See also [Cameron and Trivedi \(2005\)](#), chap. 13).

### 29.3.1 Posterior distribution

Let the data on  $N$  observations be denoted  $(\mathbf{y}, \mathbf{X})$ , where  $\mathbf{y}$  is data on the dependent variables and  $\mathbf{X}$  is data on exogenous regressors. And let  $\boldsymbol{\theta}$  denote the  $K$  parameters of the conditional density of  $\mathbf{y}$  given  $\mathbf{X}$ .

The key ingredients for Bayesian analysis are a specified likelihood function for the data and a specified prior density for the parameters. These are combined to give the posterior density as follows:

1. Likelihood function (for data given parameters), denoted  $L(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X})$ .
2. Prior density (for parameters), denoted  $\pi(\boldsymbol{\theta})$ .
3. Posterior density (for parameters given data), denoted  $p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X})$ .

As an example, we might assume that  $\mathbf{y}$ , conditional on data  $\mathbf{X}$  and parameters  $\boldsymbol{\beta}$ , is normally distributed with mean  $\mathbf{X}'\boldsymbol{\beta}$  and variance matrix  $\sigma^2\mathbf{I}$ , where  $\sigma^2$  is known. One possible prior for  $\boldsymbol{\beta}$  is that it is normally distributed with specified mean and variance. For this example, a weaker prior for  $\boldsymbol{\beta}$  has larger prior covariance matrix.

The posterior density is defined as

$$p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X}) = \frac{L(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X}) \times \pi(\boldsymbol{\theta})}{m(\mathbf{y}|\mathbf{X})} \quad (29.1)$$

where

$$m(\mathbf{y}|\mathbf{X}) = \int L(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X}) \times \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

is called the marginal likelihood. This result is obtained by repeated use of Bayes rule. First, note that  $\Pr[A|B] = \Pr[A \cap B]/\Pr[B]$  by Bayes rule. Second,  $\Pr[A \cap B] = \Pr[B|A] \times \Pr[A]$  by the multiplication rule. Combining yields  $\Pr[A|B] = \Pr[B|A] \times \Pr[A]/\Pr[B]$ . Here  $\boldsymbol{\theta}$  corresponds to  $A$ ,  $\mathbf{y}$  corresponds to  $B$ , and we have additionally conditioned on  $\mathbf{X}$  throughout.

The marginal likelihood is a normalizing constant that does not depend on  $\boldsymbol{\theta}$ , and hence the result is more simply expressed as

$$\begin{aligned} p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X}) &\propto L(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X}) \times \pi(\boldsymbol{\theta}) \\ \text{Posterior} &\propto \text{Likelihood} \times \text{Prior} \end{aligned}$$

where the symbol  $\propto$  means “is proportional to”.

The challenge in implementing the above is obtaining the normalizing constant  $m(\mathbf{y}|\mathbf{X})$  because it involves a  $K$ -dimensional integral that has no analytical solution except in some very simple models. Initial work used numerical methods such as importance sampling to compute this integral. More recently, MCMC methods provide a way to make draws from the posterior  $p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X})$  without having to compute the marginal likelihood  $m(\mathbf{y}|\mathbf{X})$ .

### 29.3.2 MCMC methods

The target object of inference is the unknown posterior distribution  $p(\boldsymbol{\theta} | \mathbf{y}, \mathbf{X})$ . MCMC methods provide a computational methodology for sequentially creating a chain that converges to the target posterior, which is called the stationary distribution. The key insight is that even if the mathematical expression of the posterior is not known, a large sample of draws from the posterior is equivalent to it. It is then straightforward to estimate the moments of the posterior, such as the posterior mean, as well as functions of estimates and data, such as marginal effects (MES) in a nonlinear regression model.

The goal then is to make draws of  $\theta$  from the posterior. One way to do so is to make draws from a proposal or candidate distribution, such as the normal, and then use a rule to decide whether to accept such draws as being draws from the target distribution, here the posterior distribution.

The simplest such method, accept–reject sampling, is not possible, because it uses an acceptance rule based on the functional form for the posterior distribution; in most cases, this is unknown because of lack of an analytical expression for the marginal likelihood  $m(\mathbf{y}|\mathbf{X})$ .

Instead, MCMC sampling methods are used. These make posterior draws, denoted  $\theta_s$  in the  $s$ th round, based on the draw  $\theta_{s-1}$  in the previous round. The term “Monte Carlo” arises because of reliance on random draws. The draws form a Markov chain because once we condition on  $\theta_{s-1}$ , the preceding draws  $\theta_{s-2}, \theta_{s-3}, \dots$  have no additional impact on  $\theta_s$ .

Under appropriate assumptions, Markov chains eventually converge to a stationary distribution, in which case subsequent draws  $\theta_{s+1}, \theta_{s+2}, \dots$  all have the same marginal distribution. Bayes MCMC methods are set up so that the stationary distribution is the desired posterior  $p(\theta|\mathbf{y}, \mathbf{X})$ .

The posterior draws are clearly correlated because  $\theta_s$  depends on  $\theta_{s-1}$ . There is nonetheless no problem in using the draws from the posterior, though if they are highly correlated, then one needs to make more draws to ensure sufficient precision.

MCMC methods potentially enable one to specify very complicated models. But it is easy to perform invalid analysis. Poorly identified or even unidentified models can yield seemingly reasonable results.

Furthermore, even in well-specified models, there is no guarantee that the Markov chain has converged to its stationary distribution. The Stata default is to discard the first 2,500 MCMC draws, called “burn-in” draws, and to retain only subsequent draws. But the chain may take many more draws than this to converge. Unfortunately, there is no formal test for whether the chain has converged; some diagnostics are presented in sections [29.4](#) and [29.6](#).

### 29.3.3 Metropolis–Hastings algorithm

The main MCMC method is the Metropolis–Hastings (MH) algorithm. This algorithm uses an acceptance rule based on a ratio of two posterior densities  $p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X})$  for which the (unknown) marginal likelihood term cancels out. Specifically, for draws  $s = 1, \dots, S$ ,

1. generate a draw  $\boldsymbol{\theta}_s^*$  from the proposal density or candidate density  $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}_{s-1})$ . For the commonly used random-walk MH algorithm, the proposal density is the normal or multivariate normal with mean  $\boldsymbol{\theta}_{s-1}$ .
2. calculate the acceptance probability to equal  $a_s$ , where

$$a_s = \min\{r(\boldsymbol{\theta}_s^*|\boldsymbol{\theta}_{s-1}), 1\}$$

$$r(\boldsymbol{\theta}_s^*|\boldsymbol{\theta}_{s-1}) = \frac{p(\boldsymbol{\theta}_s^*|\mathbf{y}, \mathbf{X}) \times q(\boldsymbol{\theta}_{s-1}|\boldsymbol{\theta}_s^*)}{p(\boldsymbol{\theta}_{s-1}|\mathbf{y}, \mathbf{X}) \times q(\boldsymbol{\theta}_s^*|\boldsymbol{\theta}_{s-1})}$$

3. draw  $u_s$  from the uniform  $(0, 1)$  distribution.
4. accept  $\boldsymbol{\theta}_s = \boldsymbol{\theta}_s^*$  if  $u_s < a_s$  and otherwise set  $\boldsymbol{\theta}_s = \boldsymbol{\theta}_{s-1}$ .

In step 2, the ratio of posteriors  $p(\boldsymbol{\theta}_s^*|\mathbf{y}, \mathbf{X})/p(\boldsymbol{\theta}_{s-1}|\mathbf{y}, \mathbf{X})$  simplifies to

$$\frac{L(\boldsymbol{\theta}_s^*|\mathbf{y}, \mathbf{X}) \times \pi(\boldsymbol{\theta}_s^*)}{L(\boldsymbol{\theta}_{s-1}|\mathbf{y}, \mathbf{X}) \times \pi(\boldsymbol{\theta}_{s-1})}$$

because the marginal likelihood  $m(\mathbf{y}|\mathbf{X})$  is a common term that does not depend on  $\boldsymbol{\theta}$  and hence cancels out in the ratio.

Further simplification is possible if the proposal density is a symmetric density, satisfying  $q(\boldsymbol{\theta}_{s-1}|\boldsymbol{\theta}_s^*) = q(\boldsymbol{\theta}_s^*|\boldsymbol{\theta}_{s-1})$ . An example is the random-walk MH algorithm with  $\boldsymbol{\theta}_s^*|\boldsymbol{\theta}_{s-1} \sim \mathcal{N}[\boldsymbol{\theta}_{s-1}, \mathbf{V}]$  for fixed  $\mathbf{V}$ . In that case, the MH algorithm is strictly speaking the Metropolis algorithm, but the more general terminology “MH algorithm” is still commonly used.

### 29.3.4 Sampling efficiency

MCMC methods lead to draws from the posterior that are correlated. As a result, 10,000 MCMC draws are equivalent to many fewer independent draws.

Let  $\hat{\rho}_j$  denote the  $j$ th autocorrelation of the posterior draws. The sampling efficiency statistic given in the `bayesmh` output equals  $1/(1 + 2 \times \sum_{j=1}^{\max} \hat{\rho}_j)$ , where `max` is the minimum of 500 and the lag at which  $|\hat{\rho}_j| < 0.01$ .

For example, if the sampling efficiency is 0.05, then 10,000 MCMC draws are equivalent to  $0.05 \times 10000 = 500$  independent draws. One way that sampling efficiency of 0.05 can arise is if  $\hat{\rho}_j = 0.95^j$ . Sampling efficiency this low is not unusual.

### 29.3.5 Gibbs sampling algorithm

The Gibbs sampler is an MCMC method that has smaller autocorrelation of the MCMC draws, leading to higher sampling efficiency. This method can be applied in special cases where analytical results are available for one or more conditional posteriors (as usual, analytical results for the unconditional posterior are not available).

Consider the case where parameters are split into two blocks, so  $\theta = (\theta^1, \theta^2)$ . If we knew both the conditional posterior of  $\theta^1 | \theta^2$  and the conditional posterior of  $\theta^2 | \theta^1$ , then we could directly apply the Gibbs sampler, alternating draws from the two conditional posteriors. Section 5.4.6 provides a Gibbs sampler example.

In practice, this is rarely possible. But in some cases, we may know one of the conditional posteriors, say, that for  $\theta^1 | \theta^2$ . Then, we draw  $\theta_s^1$  from the conditional posterior of  $\theta^1 | \theta_{s-1}^2$  and use the MH algorithm to draw  $\theta_s^2$  given  $(\theta_s^1 | \theta_{s-1}^2)$ . The manual entry for [BAYES] `bayesmh` gives combinations of likelihood and prior for which such a hybrid MH scheme is possible. An example is given in section [29.7.2](#).

### 29.3.6 Bayesian and classical approaches compared

In classical statistics, given data and the likelihood function for population parameters  $\theta$ , we obtain the MLE  $\hat{\theta}$ , which is random because of sampling error. In all but the simplest models, statistical inference on  $\theta$  is based on asymptotic approximations. A resulting 95% confidence interval for a scalar parameter  $\theta$  is interpreted as one that with repeated sampling of the population will include  $\theta$  95% of the time.

The Bayesian approach introduces randomness via prior beliefs on the value of the population parameters that is not exact. This prior information on  $\theta$  is combined with sample information to yield the posterior distribution of  $\theta$ . Given the posterior distribution, statistical inference on  $\theta$  is direct and does not require asymptotic approximations. And a 95% Bayesian credible region is directly interpreted as being an interval that scalar  $\theta$  lies in with probability 0.95.

Additionally, there are decision-theoretic underpinnings for the Bayesian approach. Suppose we seek an estimate  $\hat{\theta}$  of  $\theta$  that minimizes a loss function  $L(\theta, \hat{\theta})$ . Then the optimal estimator is the estimator that minimizes expected posterior loss, where expectation is with respect to the posterior distribution of  $\theta$ . The posterior mean is optimal given quadratic loss, and the posterior median is optimal given absolute error loss. Furthermore, the optimal Bayes estimator minimizes expected risk for a specified loss function, where expected risk averages expected posterior loss over possible samples (by additionally integrating with respect to the likelihood function).

Rather than provide a Bayesian interpretation to results, some applied statisticians and econometricians use Bayesian methods with a noninformative prior as a computational tool to obtain the MLE in settings where alternative computational methods such as maximum simulated likelihood or quadrature are not as computationally efficient. In that case, the posterior mode, the peak of the posterior density, should be used because maximizing the likelihood function corresponds to finding the mode.

The `bayes` prefix and `bayesmh` command do not report the posterior mode because it is not of interest to Bayesians and because its computation can be problematic if there are multiple peaks of the posterior such as in the density plots in figure 29.1. The posterior mean is often used instead for

convenience and is close to the posterior mode if the posterior density is unimodal and symmetric.

As the sample size gets larger, the difference between the posterior mean and the MLE narrows. For simplicity, assume that the observations are i.i.d. Then, taking the logarithm of the posterior defined in (29.1), we have

$$\sum_{i=1}^N \ln p(\boldsymbol{\theta}|y_i) \propto \ln \pi(\boldsymbol{\theta}) + \sum_{i=1}^N \ln f(y_i|\boldsymbol{\theta})$$

In a large sample, the posterior is dominated by the likelihood contribution because the contribution of the prior to the posterior remains fixed, while the contribution of the sample to the posterior grows with  $N$ . Furthermore, if the likelihood satisfies the standard regularity conditions, so that the MLE is root- $N$  consistent and asymptotically normal, it can be shown that the posterior distribution of  $\boldsymbol{\theta}$  has an asymptotic normal distribution with mean the MLE and variance the inverse of the information matrix; see, for example, [Cameron and Trivedi \(2005, 432\)](#).

In the presentation below, we contrast the posterior mean, standard deviation, and 95% credible region of a parameter with the corresponding MLE, standard error, and 95% confidence interval. Differences between the two reflect in part the informativeness of the prior and finite sample asymmetry in the posterior distribution.

### 29.3.7 The `bayesmh` command

The `bayesmh` command is the essential Stata command for Bayesian analysis. The command covers a very broad range of models, one broader than the simpler `bayes` prefix, introduced in section [29.2.1](#).

The basic syntax for this command for a univariate regression model is

```
bayesmh depvar [indepvars] [if] [in] [weight], likelihood(modelspec) prior(priorspec)
[options]
```

The likelihood can be that for leading models, including generalized linear models (normal, probit, logit, poisson, exponential), as well as lognormal and ordered probit and logit. Additionally, a user-provided log likelihood can be specified using the option `evaluator()`.

The prior distribution for a scalar parameter, discussed further below, can be normal, Student's  $t$ , Cauchy, lognormal, uniform, gamma, inverse-gamma, exponential, beta, Laplace, Pareto, chi-squared, flat, Jeffreys, Bernoulli, Poisson, geometric, or a discrete index. In the vector case, priors include the multivariate normal, Wishart, inverse-Wishart, Dirichlet, and Jeffreys and Zellner's  $g$ -prior. Additionally, a user-provided prior density can be specified. For replicability, one should set the seed using the `rseed()` option of the `bayesmh` command because results will always vary with the initial seed.

Postestimation command `bayesstats summary` provides summary statistics for functions of the parameters; commands `bayesgraph`, `bayesstats ess`, and `bayesstats grubin` present diagnostics for the MCMC draws; commands `bayesstats ic`, `bayestest interval`, and `bayestest model` are used for Bayesian inference and model selection; command `bayesstats ppvalues` is used for posterior model checks; and command `bayespredict` is used for Bayesian prediction. All but the last two of these commands are also available after the `bayes` prefix.

## 29.4 An i.i.d. example

As an example, consider analysis of i.i.d. normally distributed data with mean  $\mu$  and known variance of 100. We consider Bayesian inference on  $\mu$  given a normal prior for  $\mu$ . The analysis is quite detailed and illustrates Stata Bayesian commands that extend directly to multiple regression.

### 29.4.1 MLE

Suppose  $y_i | \mu \sim \mathcal{N}[\mu, 100]$ . We generate a sample of 50 observations that turns out to have sample mean 10.72 and standard deviation 10.90.

```
. * Generate a sample of 50 observations on y
. clear
. qui set obs 50
. set seed 10101
. gen y = rnormal(10,10)
. summarize
```

| Variable | Obs | Mean     | Std. dev. | Min       | Max     |
|----------|-----|----------|-----------|-----------|---------|
| y        | 50  | 10.71767 | 10.90006  | -21.70636 | 38.1053 |

In this example, the MLE is the sample mean, so  $\hat{\mu} = \bar{y} \simeq 10.72$ , with variance  $s/\sqrt{N} = 10.90/\sqrt{50} \simeq 1.54$ . Command `mean` yields

```
. * The MLE is the sample mean
. mean y
Mean estimation Number of obs = 50

+-----
| Mean Std. err. [95% conf. interval]
+-----
y | 10.71767 1.541502 7.61991 13.81544
+-----
```

The resulting 95% confidence interval for  $\mu$  is  $[7.62, 13.82]$ . This interval is interpreted as meaning that if we were to repeat this estimation procedure many times on many independent samples, then 95% of the time the resulting confidence interval will include the unknown constant  $\mu$ .

## 29.4.2 Bayesian analysis

We use command `bayesmh` with option `likelihood()` used to specify the likelihood and option `prior()` used to specify the prior.

Here we specify the prior to be  $\mu \sim \mathcal{N}[5, 4]$ .

```
. * Bayesian posterior for mu with normal y and N(5,4) prior for mu
. bayesmh y, likelihood(normal(100)) prior({y:_cons}, normal(5,4))
> rseed(10101) saving(mcmcdraws_iid, replace)
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:

$y \sim \text{normal}(\{y:_\text{cons}\}, 100)$

Prior:

$\{y:_\text{cons}\} \sim \text{normal}(5, 4)$

---

|                                          |                  |   |        |
|------------------------------------------|------------------|---|--------|
| Bayesian normal regression               | MCMC iterations  | = | 12,500 |
| Random-walk Metropolis-Hastings sampling | Burn-in          | = | 2,500  |
|                                          | MCMC sample size | = | 10,000 |
|                                          | Number of obs    | = | 50     |
|                                          | Acceptance rate  | = | .4332  |
| Log marginal-likelihood = -193.45168     | Efficiency       | = | .2282  |

| y     | Equal-tailed |           |        |          |                      |         |
|-------|--------------|-----------|--------|----------|----------------------|---------|
|       | Mean         | Std. dev. | MCSE   | Median   | [95% cred. interval] |         |
| _cons | 8.797346     | 1.162716  | .02434 | 8.832482 | 6.502072             | 11.0129 |

file mcmcdraws\_iid.dta not found; file saved.

The results are obtained using random-walk MH sampling. This is the MH algorithm with proposal density for  $\mu_s^*$  the  $N(\mu_{s-1}, \sigma_s^2)$  distribution, where the complicated rule to periodically update  $\sigma_s^2$  is detailed in [BAYES] **bayesmh**.

The Stata default is to discard the initial 2,500 “burn-in” draws of  $\mu$  because the MH algorithm takes time to converge to the posterior distribution. The Stata default is to retain the subsequent 10,000 draws; these are viewed as (correlated) draws from the posterior.

The acceptance rate is 0.4332, meaning that 4,332 of the 10,000 draws accept  $\mu_s = \mu_s^*$  and so are different from the immediately preceding draw, while for the remaining 5,668 draws  $\mu_s = \mu_{s-1}$ . If the acceptance rate is close to zero, then few draws are accepted, so only a small part of the posterior has been searched. If the acceptance rate is close to one, then we are drawing too often from the proposal distribution, not the posterior distribution. For the random-walk MH algorithm, studies suggest that the optimal acceptance rate is a bit below 0.5 for the univariate parameter case and a bit below 0.25 in the multiparameter case. So an acceptance rate of 0.4432 is very good.

The more important efficiency statistic of 0.2282 means that the 10,000 correlated draws provide the same information content as 2,282 independent draws from the posterior. This loss in efficiency arises both because only 4,332 draws are accepted and because even these accepted draws are correlated. An efficiency of 0.22 is regarded as very good.

From the results section of the Stata output, the posterior mean is 8.80 and the posterior standard deviation is 1.16. The posterior mean lies between the sample MLE  $\hat{\mu} = \bar{y} \simeq 10.72$  and the prior mean of  $\mu$ , which equaled 5. The posterior standard deviation of 1.16 is less than the standard error of  $\hat{\mu} \simeq 1.54$ .

This result accords with intuition. The posterior mean of a parameter lies between the sample MLE and the prior mean, and prior information reduces variability so that the posterior variance of the parameter is less than the variance of the MLE of the parameter. In some standard models, this result always holds; see section [29.4.8](#) for analytical results for this example.

### 29.4.3 Bayesian inference

The equal-tailed Bayesian credible region, also given in the preceding Stata output, lies between the 2.5 and 97.5 percentiles of the 10,000 MCMC posterior draws of  $\mu$ . It is directly interpreted as saying that  $\mu$  lies between 6.50 and 11.01 with probability 0.95. This interval is narrower than the 95% confidence interval for the MLE of [7.62, 13.82] because prior information reduces variability. And, as already noted, the interpretation of the two intervals is quite different.

The posterior probability that  $\mu$  lies in a certain range is simply the fraction of the 10,000 MCMC draws that lie in the given range.

For example, command `bayestest interval` finds that the probability that  $\mu$  exceeds 10 equals 0.1423 because

```

. * Bayesian hypothesis test: Pr[mu > 10]
. bayestest interval {y:_cons}, lower(10)
Interval tests MCMC sample size = 10,000
 prob1 : {y:_cons} > 10

```

---

|       | Mean  | Std. dev. | MCSE     |
|-------|-------|-----------|----------|
| prob1 | .1423 | 0.34938   | .0066141 |

Inference on transformations of the parameters is also straightforward. For example, the posterior distribution of  $\mu^2$  is obtained by simply squaring each of the 10,000 MCMC draws of  $\mu$ . This is done using command

```

. * Bayesian statistics for transformation of parameter mu
. bayesstats summary ({y:_cons}^2)

Posterior summary statistics MCMC sample size = 10,000
 expr1 : {y:_cons}^2


```

---

|       | Mean     | Std. dev. | MCSE    | Median   | Equal-tailed<br>[95% cred. interval] |         |
|-------|----------|-----------|---------|----------|--------------------------------------|---------|
| expr1 | 78.74506 | 20.48997  | .426352 | 78.01275 | 42.27698                             | 121.284 |

In this example, all draws of  $\mu$  are positive. It follows that the  $q$ th percentile of  $\mu^2$  equals the square of the  $q$ th percentile of  $\mu$ . For example, the median of  $\mu$  is 8.83248, and the median of  $\mu^2$  is  $8.83248^2 = 78.0128$ . Similarly, given that the 95% credible region for  $\mu^2$  is equal to [6.5021, 11.0129], the 95% credible region for  $\mu^2$  is equal to [6.5021<sup>2</sup>, 11.0129<sup>2</sup>] or [42.27, 121.28].

Note that if instead we used the MLE, then we would use the delta method, which relies on asymptotic theory. Because  $\partial\mu^2/\partial\mu = 2\mu$ , and  $\hat{\mu} = \bar{y}$ , the standard error of  $\hat{\mu}^2 = \bar{y}^2 = 10.718^2 = 114.88$  equals

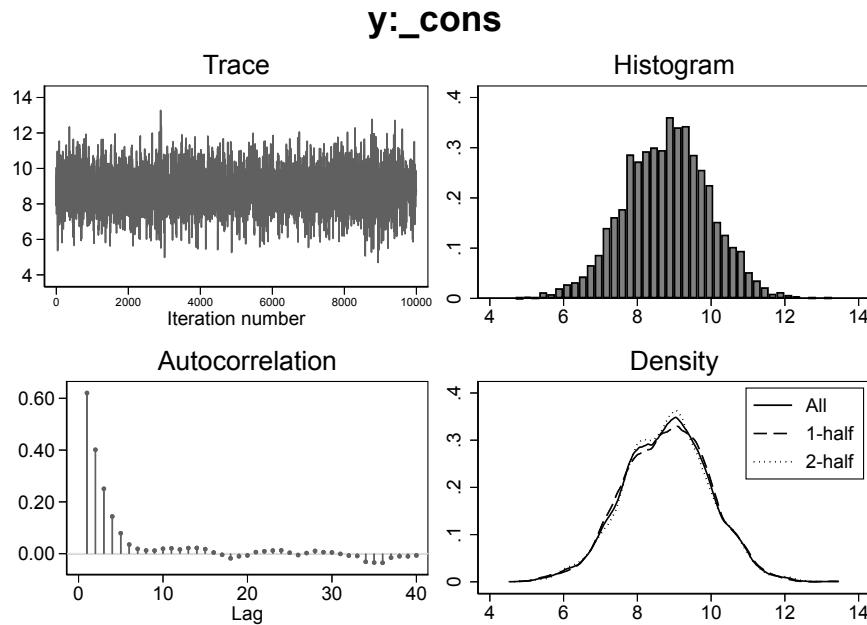
$2\bar{y} \times \text{se}(\bar{y}) = 2 \times 10.718 \times 1.541 = 33.04$ , and an asymptotic 95% confidence interval for  $\mu^2$  is  $114.88 \pm 1.96 \times 33.04 = [50.12, 179.64]$ .

#### 29.4.4 MCMC diagnostics

It is absolutely essential to gauge whether the Markov chain has converged and to measure the computational efficiency of the MH algorithm. However, there is no formal test that confirms that the algorithm has converged.

Several useful graphs can be obtained using command `bayesgraph diagnostics`. We obtain

- . \* Diagnostic plots for MH posterior draws
- . `bayesgraph diagnostics {y:_cons}, scale(1.1)`



**Figure 29.2.** Diagnostics for 10,000 MH posterior draws of  $\mu$

The top left panel of figure 29.2 plots the 10,000 sequential MH draws of  $\mu$ . Red flags include a trend or cycle in the draws and little change from draw to draw; there is no problem here.

The top right panel of figure 29.2 gives a histogram of the draws. Care is needed if this is multimodal because it may indicate that the chain has not

converged but is instead getting trapped in subareas of the posterior.

The bottom left panel of figure 29.2 shows the autocorrelations of the draws. These should die out quickly, the case here. The efficiency statistic of 0.2282 given in the `bayesmh` output equals  $1 / \left(1 + 2 \times \sum_{j=1}^{\max} \hat{\rho}_j\right)$ , where  $\hat{\rho}_j$  is the  $j$ th autocorrelation of the posterior draws and max is the minimum of 500 and the lag at which  $|\hat{\rho}_j| < 0.01$ .

If the posterior draws were independent, then the Monte Carlo simulation error, reported in output as `MCSE`, would equal the posterior standard deviation  $s_\theta$  divided by  $\sqrt{S}$ , where  $S$  is the number of MCMC draws. In fact, the draws are correlated, and the larger measure  $s_\theta / \sqrt{S/\text{eff}}$  is used where `eff` is the efficiency statistic.

The bottom right panel of figure 29.2 shows the kernel density estimate for all 10,000 draws, a smoothed version of the histogram in the top right panel. If the chain has converged, then the dashed kernel density estimate for the first 5,000 draws should be very similar to the dotted estimate for the second 5,000 draws. At the same time, however, similar density estimates for the two halves do not guarantee convergence.

All four panels suggest that the chain has converged and is relatively efficient.

#### 29.4.5 MCMC diagnostics using different chains

A useful diagnostic method for chain convergence is to use multiple chains. If the posterior differs substantially across chains, then the chain has not converged. The test can be made more stringent by using quite different, and potentially quite extreme, starting values for each chain.

The `nchains()` option of the `bayesmh` command or the `bayes` prefix enables estimation of multiple chains.

Before using the `nchains()` option, we manually fit 4 distinct chains with initial values that are random draws from  $\mathcal{N}(10, 6^2)$  that should give a

wide range of starting values because the MLE for  $\mu$  was 10.72 with standard deviation 1.54.

```
. * Manually obtain posterior means and st. devs for four different chains
. set seed 10101
. forvalues i=1/4 {
 2. local start = rnormal(10,6^2)
 3. quietly bayesmh y, likelihood(normal(100))
> prior({y:_cons}, normal(5,4))
> initial({y:_cons} `start')
 4. matrix pstart = (nullmat(pstart) \ `start')
 5. matrix pmeans = (nullmat(pmeans) \ e(mean))
 6. matrix psds = (nullmat(psds) \ e(sd))
 7. }
. matrix p_all = pstart,pmeans,psds
. matrix list p_all, title("Start value, post. means and st. devs. for 4 chains")
p_all[4,3]: Start value, post. means and st. devs. for 4 chains
 y: y:
 c1 _cons _cons
r1 24.119756 8.8182371 1.1682963
r2 7.6265723 8.8136099 1.1483331
r3 11.02609 8.8414756 1.1638707
r4 -23.910821 8.8207744 1.1702667
```

The four starting values are very different but lead to very similar posterior means and standard deviations for the parameter. The function `nullmat()` is used because the matrix `pstart`, for example, is not previously defined at the first pass through the loop.

The following code computes both the between variation (across the 4 starting values) and the within variation (across the 10,000 MH draws for a given starting value) for the posterior mean.

```

. * Compute within and between variation across the 4 MCMC runs
. mata
----- mata (type end to exit) -----
: pmeans = st_matrix("pmeans")
: psd = st_matrix("psds")
: W = (psd`psd)/4
: meanpmeans = mean(pmeans)
: B = (pmeans-meanpmeans*J(4,1,1))`*(pmeans-meanpmeans*J(4,1,1))/(4-1)
: // (1) Between, (2) within, (3) Between/total, (4) Total / Within
: (B, W, B/(B+W), (B+W)/W)
: 1 2 3 4
1 .0001520204 1.351926088 .0001124346 1.000112447
: end

```

The variation between the four MH runs is very small relative to the within variation, as desired. Similar to the other diagnostics, failing this test signals nonconvergence of the MH algorithm, but passing the test does not guarantee convergence of the chain.

The `nchains()` option of the `bayesmh` command or `bayes` prefix automates the preceding process. Using four chains, we obtain

```

. * Four different chains using nchains() option
. bayesmh y, nchains(4) likelihood(normal(100)) rseed(10101)
> prior({y:_cons}, normal(5,4)) init1({y:_cons} 24)
> init2({y:_cons} 7) init3({y:_cons} 11)
> init4({y:_cons} -23)
> initsummary nomodelsummary
Chain 1
 Burn-in ...
 Simulation ...
Chain 2
 Burn-in ...
 Simulation ...
Chain 3
 Burn-in ...
 Simulation ...
Chain 4
 Burn-in ...
 Simulation ...

Initial values:
Chain 1: {y:_cons} 24
Chain 2: {y:_cons} 7
Chain 3: {y:_cons} 11
Chain 4: {y:_cons} -23

Bayesian normal regression
Random-walk Metropolis-Hastings sampling
Number of chains = 4
Per MCMC chain:
Iterations = 12,500
Burn-in = 2,500
Sample size = 10,000
Number of obs = 50
Avg acceptance rate = .4383
Avg efficiency = .2314
Max Gelman-Rubin RC = 1
Avg log marginal-likelihood = -193.45686

```

|       | Equal-tailed<br>[95% cred. interval] |           |         |          |                   |
|-------|--------------------------------------|-----------|---------|----------|-------------------|
| y     | Mean                                 | Std. dev. | MCSE    | Median   |                   |
| _cons | 8.812606                             | 1.156895  | .012025 | 8.816636 | 6.540255 11.07231 |

Note that the `rseed()` option used here ensures that a different seed is used for each chain. The default uses parameter starting values that do not vary greatly across each chain. Here the `init1()` to `init4()` options are used to specify a wider range of starting values.

The output includes the Gelman–Rubin  $R_C$  statistic. Let  $\hat{\theta}_j$  and  $s_j^2$  be the posterior mean and variance in the  $j$ th chain,  $\hat{\theta} = 1/m \sum_{j=1}^m \hat{\theta}_j$ ,

$B = 1/(m - 1) \sum_{j=1}^m (\hat{\theta}_j - \bar{\theta})^2$  measure the variation between chains, and  $W = 1/m \sum_{j=1}^m s_j^2$  denote the average variation within each chain. A crude measure of chain convergence is small between variation relative to within variation, in which case  $(W + B)/W$  is close to one. The  $R_c$  statistic is a refined version of this measure that adjusts for a finite number of chains. A common threshold for chain convergence is that  $R_c < 1.1$ . Because  $R_c = 1$ , upon rounding, we conclude the chain has converged.

The preceding output includes the maximum value across all parameters of the  $R_c$  statistic. The `bayesstats grubin` command provides the  $R_c$  statistic for each parameter.

```
. bayesstats grubin
Gelman-Rubin convergence diagnostic
Number of chains = 4
MCMC size, per chain = 10,000
Max Gelman-Rubin Rc = 1.000105
```

|       | Rc       |
|-------|----------|
| y     |          |
| _cons | 1.000105 |

Convergence rule:  $R_c < 1.1$

Here there is only one parameter and  $R_c = 1.000105$ , which equals 1 when rounded.

#### 29.4.6 Sensitivity analysis

One should also check the sensitivity of results to the specification of the prior, where the priors should be consistent with prior information and potentially use different distributions.

As an example, we consider a  $N(4, 8)$  prior and a  $\chi^2(4)$  prior. Both priors have mean 4 and variance 8 but differ because the chi-squared distribution is asymmetric and takes positive values only. We have

```

. * Compare posterior for two different priors
. quietly bayesmh y, likelihood(normal(100)) prior({y:_cons}, normal(4,8))
> rseed(10101)
. matrix pmeans = e(mean)
. matrix psds = e(sd)
. qui bayesmh y, likelihood(normal(100)) prior({y:_cons}, chi2(4))
> rseed(10101)
. matrix pmeans = pmeans \ e(mean)
. matrix psds = psds \ e(sd)
. matrix p_all = pmeans,psds
. matrix list p_all, title("Post. means and st. devs. for 2 different priors")
p_all[2,2]: Post. means and st. devs. for 2 different priors
 y: y:
 _cons _cons
Mean 9.3516909 1.2641803
Mean 9.9058014 1.4079165

```

The posterior means are within half a posterior standard deviation of each other.

#### 29.4.7 Further analysis of the draws

The results of the original use of the `bayesmh` command in this section were saved in file `mcmcdraws_iid.dta`. This file contains the 4,332 unique draws of  $\mu_s$ , stored in variable `eq1_p1` (for equation 1 parameter 1), along with the number of consecutive times that each unique draw was obtained, stored in variable `_frequency`.

```

. * Summarize the unique retained draws
. use mcmcdraws_iid, clear
. summarize

```

| Variable          | Obs   | Mean      | Std. dev. | Min       | Max       |
|-------------------|-------|-----------|-----------|-----------|-----------|
| _chain            | 4,332 | 1         | 0         | 1         | 1         |
| _index            | 4,332 | 5018.283  | 2898.168  | 1         | 9998      |
| _loglikeli^d      | 4,332 | -191.4893 | 1.30697   | -199.2417 | -190.185  |
| _logposterr       | 4,332 | -195.1071 | .7662066  | -201.9883 | -194.5214 |
| eq1_p1            | 4,332 | 8.805722  | 1.249882  | 4.698787  | 13.27406  |
| <u>_frequency</u> | 4,332 | 2.308403  | 1.771655  | 1         | 14        |

The mean and standard deviation of variable `eq1_p1` differ from the previously reported posterior mean and posterior standard deviation of, respectively, 8.797 and 1.163. This is because the posterior mean and standard deviation given in the `bayesmh` output are computed using frequency weights for the repeated values.

All 10,000 MCMC draws, including the repeated draws, can be obtained by expanding the dataset and sorting by variable `_index`. Summarizing the complete dataset yields

```
. * Expand to get the 10,000 MH draws, including repeated draws
. expand _frequency
(5,668 observations created)

. sort _index

. gen s = _n

. summarize eq1_p1
```

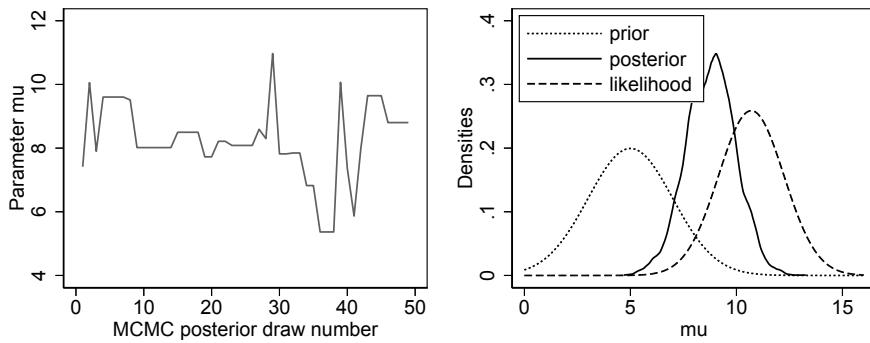
| Variable | Obs    | Mean     | Std. dev. | Min      | Max      |
|----------|--------|----------|-----------|----------|----------|
| eq1_p1   | 10,000 | 8.797346 | 1.162716  | 4.698787 | 13.27406 |

The variable `eq1_p1` contains the 10,000 draws of  $\mu$ , including repeated values. This variable has mean 8.797346 and standard deviation 1.162716, which exactly equal the posterior mean and standard deviation in the output given after command `bayesmh`.

Given all 10,000 draws of  $\mu$ , we can reproduce the various graphs obtained earlier using command `bayesgraph`. For example, the following code obtains the trace for just the first 50 MCMC draws.

```
. * Graph the first 50 draws of mu
. quietly tsset s
. tsline eq1_p1 if s < 50, scale(1.5) ytitle("Parameter mu")
> xtitle("MCMC posterior draw number")
```

The left panel of figure 29.3 shows the first 50 posterior draws of  $\mu$ . The flat sections are cases where the draw  $\mu_s^*$  was not accepted and instead  $\mu_s = \mu_{s-1}$ .



**Figure 29.3.** Diagnostics for 10,000 MH posterior draws of  $\mu$

The right panel of figure 29.3, produced by Stata code not given here, plots the normal likelihood, the normal prior density, and the kernel density estimate of the posterior. It is clear that the posterior mean lies between the sample mean and the prior mean and that the posterior variance is less than either the sample or prior variance.

#### 29.4.8 Analytical results

In the current example, it is actually possible to obtain a tractable solution for the posterior, in which case the MCMC methods are not necessary.

Suppose  $y_i|\mu \sim N(\mu, \sigma^2)$ , where  $\sigma^2$  is known. Then for  $\mathbf{y} = (y_1, \dots, y_N)$  and independent data, the likelihood is  $L(\mathbf{y}|\mu) = [\{1/(2\pi\sigma^2)\}]^{N/2} \exp[-\{1/(2\sigma^2)\} \sum_{i=1}^N (y_i - \mu)^2]$ . And suppose the prior for  $\mu$  is  $\mu \sim N(\underline{\mu}, \underline{s}^2)$ , where values for  $\underline{\mu}$  and  $\underline{s}^2$  are specified and an underscore is used to denote the values of parameters in the prior. Then the prior density is  $\pi(\mu) = (1/\sqrt{2\pi\underline{s}^2}) \exp \left\{ -\{1/(2\underline{s}^2)\}(\mu - \underline{\mu})^2 \right\}$ .

Some very considerable algebra given in [Cameron and Trivedi \(2005, chap. 13.2.2\)](#), for example, shows that the posterior distribution for  $\mu$  given  $\mathbf{y}$  is then the normal with

$$\begin{aligned}\mu | \mathbf{y} &\sim \mathcal{N}(\bar{\mu}, \bar{s}^2) \\ \bar{\mu} &= \bar{s}^2 \times \left\{ \left( \frac{\sigma^2}{N} \right)^{-1} \bar{y} + (\underline{s}^2)^{-1} \underline{\mu} \right\} \\ \bar{s}^2 &= \left\{ \left( \frac{\sigma^2}{N} \right)^{-1} + (\underline{s}^2)^{-1} \right\}^{-1}\end{aligned}$$

where an overscore is used to denote the values of parameters in the posterior.

The Bayesian terminology is to call the inverse of the variance the precision. So the posterior precision of  $\mu$ ,  $\bar{s}^{-2}$  defined above, is the sum of the sample precision of  $\bar{y}$  and the prior precision of  $\mu$ . The posterior mean,  $\bar{\mu}$  defined above, is a weighted average of the sample mean and the prior mean, where the respective weights are the sample and prior precisions.

In our example,  $\bar{y} = 10.72$ ,  $\sigma^2 = 100$ ,  $N = 50$ , and the prior sets  $\underline{\mu} = 5$  and  $\underline{s}^2 = 4$ . This yields  $\bar{s}^2 = 4/3 = 1.33$  and  $\bar{\mu} = 8.81$ , close to the values  $1.163^2 = 1.35$  and  $8.80$  obtained from the MH draws from the posterior.

In general, as the sample gets large, the information from the sample increases, while the information from the prior is unchanging, so the Bayesian posterior mean for  $\mu$  will collapse on  $\bar{y}$ , the MLE for  $\mu$ . For this example, this is clear from rewriting the posterior variance as  
 $\bar{s}^2 = (\sigma^2/N)/\{1 + (\sigma^2/N\underline{s}^2)\} \rightarrow \sigma^2/N$  as  $N \rightarrow \infty$  and  
 $\bar{\mu} = \{\bar{y} + (\sigma^2 \underline{\mu}/N\underline{s}^2)\}/\{1 + (\sigma^2/N\underline{s}^2)\} \rightarrow \bar{y}$  as  $N \rightarrow \infty$ .

By the same algebra, it is clear that as the prior variance gets large, so  $\underline{s}^2 \rightarrow \infty$ , we again obtain  $\bar{s}^2 \rightarrow (\sigma^2/N)$  and  $\bar{\mu} \rightarrow \bar{y}$ . With a weak prior, the Bayesian posterior mean goes to the MLE. This example illustrates the more general result that for Bayesian analysis asymptotically (as sample size goes to infinity), the likelihood dominates, and the prior has little impact on the posterior.

## 29.5 Linear regression

We now consider extension to linear regression under normality. A brief review is given of results given in many Bayesian texts and in [Cameron and Trivedi \(2005, chap. 13\)](#).

### 29.5.1 Normal regression with variance known

The preceding normal-normal result for i.i.d. data extends to the linear regression under normality with normal prior for  $\beta$  and  $\text{Var}(\mathbf{y}|\mathbf{X}) = \sigma^2 \mathbf{I}$  and  $\sigma^2$  known. Then the likelihood is  $\mathbf{y}|\beta, \mathbf{X} \sim \mathcal{N}(\mathbf{X}\beta, \sigma^2 \mathbf{I})$ , the prior is  $\beta \sim \mathcal{N}(\underline{\beta}, \underline{\mathbf{V}})$  and the posterior is  $\beta|\mathbf{y}, \mathbf{X} \sim \mathcal{N}(\bar{\beta}, \bar{\mathbf{V}})$ , where  $\bar{\beta}$  is a matrix-weighted sum of the MLE (equal to OLS) and the posterior mean and  $\bar{\mathbf{V}}$  is the inverse of the sum of the precision of the MLE and the prior precision.

Specifically, the posterior variance and mean of  $\beta$  are, respectively,

$$\begin{aligned}\bar{\mathbf{V}} &= [\{\sigma^2(\mathbf{X}'\mathbf{X})^{-1}\}^{-1} + \underline{\mathbf{V}}^{-1}]^{-1} \\ \bar{\beta} &= \bar{\mathbf{V}}[\{\sigma^2(\mathbf{X}'\mathbf{X})^{-1}\}^{-1} \hat{\beta}_{\text{OLS}} + \underline{\mathbf{V}}^{-1} \underline{\beta}]\end{aligned}$$

This result is of little practical use because in general  $\sigma^2$  is unknown.

### 29.5.2 Prior distributions for $\beta$ and $\sigma^2$

In the more usual case of unknown error variance  $\sigma^2$ , and possibly even unknown error variance matrix  $\Sigma$ , there are several possible choices for the priors for  $\beta$  and  $\sigma^2$  that we now summarize. These lead to different posterior distributions.

A noninformative prior (or diffuse prior or weak prior) is one that has little impact on the posterior distribution. An informative prior is one that does have an impact.

The simplest choice is to choose independent flat or constant priors for  $\beta$  and  $\sigma^2$ . Thus, we assume that  $\pi(\beta) = 1$  and  $\pi(\sigma^2) = 1$ . This seems a poor choice because then the prior is an improper density: it does not integrate to one given that  $\beta$  and  $\sigma^2$  are unbounded. Nonetheless, it leads to a proper posterior density in many leading examples, due to cancellation in the numerator and denominator of the expression for the posterior density  $p(\theta|y, X)$  defined in (29.1). However, care is then needed in using Bayesian model-selection methods, detailed in section 29.9, that are based on the marginal likelihood.

A flat prior is an example of a noninformative prior. A weakness is that it is not invariant to parameter transformation. For example, a constant prior for  $\sigma$  differs from a constant prior for  $\sigma^2$ .

Jeffreys prior, by contrast, is a noninformative prior that is invariant to parameter transformation. It sets  $\pi(\theta)$  proportional to  $[\det\{I(\theta)\}]^{1/2}$ , where  $\det\{I(\theta)\}$  is the determinant of  $I(\theta) = -E(\partial^2 \ln L / \partial \theta \partial \theta')$ . For the linear model under normality, the Jeffreys priors are  $\pi(\beta) \propto 1$  and  $\pi(\sigma^2) \propto 1/\sigma^2$ .

Conjugate priors are priors that lead to tractable results, with posterior in the same family of distributions as the prior. A natural conjugate prior is one for which additionally the likelihood is in the same family as the prior. In that case, the prior can be interpreted as providing additional data. Natural conjugate priors exist for densities in the exponential family.

For the normal model with  $\sigma^2$  known, the natural conjugate prior for  $\beta$  is the normal and leads to a normal posterior, a result given in the previous subsection.

For the normal model with  $\sigma^2$  unknown, the conjugate prior for the precision parameter  $1/\sigma^2$  is the gamma; equivalently, the conjugate prior for  $\sigma^2$  is the inverse-gamma.

Stata uses the shape-scale parameterization of the inverse-gamma, denoted  $z \sim \mathcal{G}[a, b]$ , where  $a$  is the shape parameter and  $b$  is the scale parameter. Then  $z$  has mean  $ab$  (for  $a > 1$ ) and variance  $a/b$  (for  $a > 2$ ). A

prior for  $\sigma^2$  that is inverse-gamma with shape parameter  $\nu_0/2$  and scale parameter  $\sigma_0^2\nu_0/2$  can be viewed as adding  $\nu_0$  extra observations with average error variance  $\sigma_0^2$ ; see, for example, [Gelman et al. \(2013\)](#). Then the scale parameter is  $\sigma_0^2$  times the shape parameter, and smaller values of  $\nu_0$  correspond to a weaker prior.

More generally, when  $\text{Var}(\mathbf{y}) = \Sigma$ , rather than  $\sigma^2\mathbf{I}$ , a Wishart prior for  $\Sigma^{-1}$  can be used. This is a matrix generalization of using the gamma prior for  $1/\sigma^2$ .

Conjugate priors are examples of informative priors. They can be made vague or diffuse or flat or noninformative by specifying a large variance in the prior. Like the flat prior, the resulting posterior is not invariant to reparameterization. Hierarchical priors are priors whose parameters in turn are random, depending on parameters for which a prior is specified. These are presented in section [29.8.2](#).

It is clear that a wide range of prior distributions might be used, unless analysis is being conducted in a setting where prior beliefs suggest a particular functional form for the prior. A range of priors might be used to determine the sensitivity of results to the choice of prior.

### 29.5.3 Posterior densities for normal regression with variance unknown

Given modern MCMC methods, there is no need to restrict attention to priors that lead to tractable results for the posterior. Thus, we might routinely specify normal priors or multivariate Student  $t$  priors for parameters in  $(-\infty, \infty)$ , gamma or lognormal priors for parameters in  $(0, \infty)$ , and beta priors for parameters in  $(0, 1)$ , regardless of the specified likelihood. And flat priors, Jeffreys priors, or priors with large variances may be used for parameters for which there is little prior knowledge.

At the same time, for more complicated models, it can be challenging to get the MH algorithm to converge, and analytical results, even for a subcomponent of the posterior, can greatly improve computational efficiency; section [29.7.1](#) provides an example. In such cases, known

tractable results for the linear regression model under normality are often used.

For the normal model with independent Jeffreys priors, we have  $\pi(\beta, \sigma^2) = \pi(\beta) \times \pi(\sigma^2) \propto 1/\sigma^2$ . Then  $p(\beta|y, X)$ , the marginal posterior for  $\beta$ , can be shown to be the multivariate Student  $t$  distribution centered at  $\hat{\beta}_{OLS}$  with  $(N - K)$  degrees of freedom and covariance matrix  $s^2(X'X)^{-1}$  multiplied by  $(N - K)/(N - K - 2)$ .

A tractable result with informative priors is obtained by specifying that the prior for  $\beta$  given  $\sigma^2$  is the normal and the prior for  $\sigma^2$  is the inverse-gamma. Then  $\pi(\beta, \sigma^2) = \pi(\beta|\sigma^2) \times \pi(\sigma^2)$ . The resulting joint posterior for  $\beta$  and  $\sigma^2$  is of similar normal inverse-gamma form. The conditional posterior for  $\beta|\sigma^2$  has mean that is a matrix weighted average of  $\hat{\beta}_{OLS}$  and the prior mean. The marginal posterior for  $\beta$  is a multivariate Student  $t$  distribution centered at the posterior mean of  $\beta$ .

It is simplest to assume that priors for the components of  $\beta$  are independent, but this is rather ad hoc. Zellner's  $g$ -prior introduces correlation by supposing that  $\beta|\sigma^2$  is normally distributed with mean  $\underline{\beta}$  and variance matrix  $g \times \sigma^2(X'X)^{-1}$ , proportional to the variance matrix of the MLE for  $\beta$ . Larger specified values for  $g$  lead to a weaker prior.

## 29.6 A linear regression example

We continue the linear regression for the log earnings example introduced in section [29.2](#). We suppose that interest lies in the slope coefficient for education and there is prior information on this parameter. For illustrative purposes, an informative prior is used for all model parameters. In practice, the Jeffreys prior or flat priors or parametric priors with large variances might be used for parameters for which there is little prior information. The sample is small, with 100 observations. Then prior information, if reasonably tight, should not be completely dominated by sample information.

### 29.6.1 MLE

We consider linear regression of log-earnings on an intercept, education, and age. The MLE for  $\beta$  when errors are i.i.d. normal and homoskedastic is simply the OLS estimator. We have

|                                                             |             |           |            |               |                      |          |
|-------------------------------------------------------------|-------------|-----------|------------|---------------|----------------------|----------|
| . * MLE for the regression (same as OLS with i.i.d. errors) |             |           |            |               |                      |          |
| . qui use mus229acs, clear                                  |             |           |            |               |                      |          |
| . quietly keep if _n <= 100                                 |             |           |            |               |                      |          |
| . regress lnearnings education age                          |             |           |            |               |                      |          |
| Source                                                      | SS          | df        | MS         | Number of obs | =                    | 100      |
| Model                                                       | 7.03491807  | 2         | 3.51745904 | F(2, 97)      | =                    | 7.52     |
| Residual                                                    | 45.3428497  | 97        | .467452059 | Prob > F      | =                    | 0.0009   |
| Total                                                       | 52.3777678  | 99        | .529068361 | R-squared     | =                    | 0.1343   |
|                                                             |             |           |            | Adj R-squared | =                    | 0.1165   |
|                                                             |             |           |            | Root MSE      | =                    | .6837    |
| lnearnings                                                  | Coefficient | Std. err. | t          | P> t          | [95% conf. interval] |          |
| education                                                   | .0852959    | .0221804  | 3.85       | 0.000         | .0412739             | .1293178 |
| age                                                         | .0079952    | .0064063  | 1.25       | 0.215         | -.0047195            | .02071   |
| _cons                                                       | 9.246449    | .4546021  | 20.34      | 0.000         | 8.34419              | 10.14871 |

The returns to an additional year of schooling in this example are 8.5% with standard error 2.2%. Earnings increase by 0.8% for each additional year of age, after controlling for education, though the coefficient is statistically insignificant at the 5% level.

## 29.6.2 Specifying the prior

For Bayesian estimation, the likelihood is specified to be the normal with independent homoskedastic errors. This is a reasonable distributional assumption for log-earnings.

As already noted, in this example, we try to use informative priors for  $\beta$  and  $\sigma^2$  wherever possible.

For the coefficient of `education`, we use a prior that is consistent with the belief that with probability 0.95 earnings rise by between 4% and 8% for each additional year of education. Then an obvious prior for the education coefficient is the normal with mean 0.06 and standard deviation 0.01 because the probability of being within two standard deviations of the mean is 0.95 for a normally distributed random variable. So the prior for  $\beta_{\text{ed}}$  is  $N(0.06, 0.01^2)$ .

For `age`, we might believe that with an additional year of aging, earnings rise by between 0% and 4% with probability 0.95, so the prior for  $\beta_{\text{age}}$  is  $N(0.02, 0.01^2)$ .

For the intercept, we really have no prior information. The simplest approach is to specify a flat prior (here also a Jeffreys prior). An alternative is a normal prior with large variance, but great care is needed to ensure the variance is large enough to guard against a poor choice of prior mean, yet not so large that the MH algorithm performs poorly. For example, a prior for  $\beta_{\text{intercept}}$  of  $N(0.00, 1.00)$  is a poor choice given that the MLE of  $\beta_{\text{intercept}}$  was 9.24. Here the prior for  $\beta_{\text{intercept}}$  is specified to be  $N(10, 10^2)$ . For the error variance  $\sigma^2$ , we use an inverse-gamma prior, for illustrative purposes, though a Jeffreys prior or a flat prior would be simpler. If we believe that 95% of individuals have earnings between \$10,000 and \$200,000, then 95% have log-earnings between 9.20 and 12.20. Assuming lognormality, this suggests a standard deviation of  $(9.20 - 12.20)/4 = 0.75$  and hence a variance of approximately 0.5 in an intercept-only model. As presented in section [29.5.2](#), if we view the inverse-gamma prior for  $\sigma^2$  as adding  $\nu_0$  extra observations with average error variance  $\sigma_0^2$ , then the prior for  $\sigma^2$  is inverse-gamma with shape parameter  $\nu_0/2$  and scale parameter  $\sigma_0^2\nu_0/2$ . Here we use

a weak prior and set  $\nu_0/2 = 0.1$  and use the inverse-gamma with parameters 1 and 0.5.

### 29.6.3 Bayesian analysis

Command `bayesmh` with the aforementioned options yields the following results:

```
. * Bayesian posterior with informative priors: Normal for b, inv gamma for s2
. bayesmh lnearnings education age, likelihood(normal({var}))
> prior({lnearnings:education}, normal(0.06,0.0001))
> prior({lnearnings:age}, normal(0.02,0.0001))
> prior({lnearnings:_cons}, normal(10,100))
> prior({var}, igamma(1,0.5))
> rseed(10101) saving(mcmcdraws_fullregress, replace)
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
  `lnearnings ~ normal(xb_lnearnings,{var})`

Priors:  
  `{lnearnings:education} ~ normal(0.06,0.0001)` (1)  
  `{lnearnings:age} ~ normal(0.02,0.0001)` (1)  
  `{lnearnings:_cons} ~ normal(10,100)` (1)  
  `{var} ~ igamma(1,0.5)`

---

(1) Parameters are elements of the linear form xb\_lnearnings.

|                                          |                    |        |
|------------------------------------------|--------------------|--------|
| Bayesian normal regression               | MCMC iterations =  | 12,500 |
| Random-walk Metropolis-Hastings sampling | Burn-in =          | 2,500  |
|                                          | MCMC sample size = | 10,000 |
|                                          | Number of obs =    | 100    |
|                                          | Acceptance rate =  | .1958  |
|                                          | Efficiency: min =  | .05397 |
|                                          | avg =              | .06435 |
| Log marginal-likelihood = -111.28922     | max =              | .08116 |

|            | Mean     | Std. dev. | MCSE    | Median   | Equal-tailed<br>[95% cred. interval] |          |
|------------|----------|-----------|---------|----------|--------------------------------------|----------|
| lnearnings |          |           |         |          |                                      |          |
| education  | .0647151 | .0088105  | .000379 | .0647482 | .0475785                             | .0824178 |
| age        | .0108165 | .0053098  | .000186 | .0111341 | .0001859                             | .0210965 |
| _cons      | 9.404009 | .2773838  | .010749 | 9.398989 | 8.869306                             | 9.980771 |
| var        | .4810144 | .0691491  | .002931 | .4731433 | .3637152                             | .6401572 |

```
file mcmcdraws_fullregress.dta not found; file saved.
. estimates store fullregress
```

The posterior means for `education` and `age` are, respectively, 0.065 and 0.011 compared with the ML estimates of 0.085 and 0.008. The corresponding posterior standard deviations are 0.0088 and 0.0053 compared with the ML standard errors of 0.0221 and 0.0064, so the informative prior for `education` in this example greatly improved precision. The posterior mean of  $\sigma^2$  is 0.481 compared with  $s^2 = 0.684^2 = 0.468$  using the reported root MSE in the output from command `regress`.

The average sampling efficiency across the four parameters is 0.0644. Command `bayesstats ess` provides the efficiency of the MH draws for each parameter.

```

. * MCMC statistics for all parameters
. bayesstats ess

Efficiency summaries MCMC sample size = 10,000
 Efficiency: min = .05397
 avg = .06435
 max = .08116

```

|            | ESS    | Corr. time | Efficiency |
|------------|--------|------------|------------|
| lnearnings |        |            |            |
| education  | 539.68 | 18.53      | 0.0540     |
| age        | 811.58 | 12.32      | 0.0812     |
| _cons      | 665.97 | 15.02      | 0.0666     |
| var        | 556.64 | 17.97      | 0.0557     |

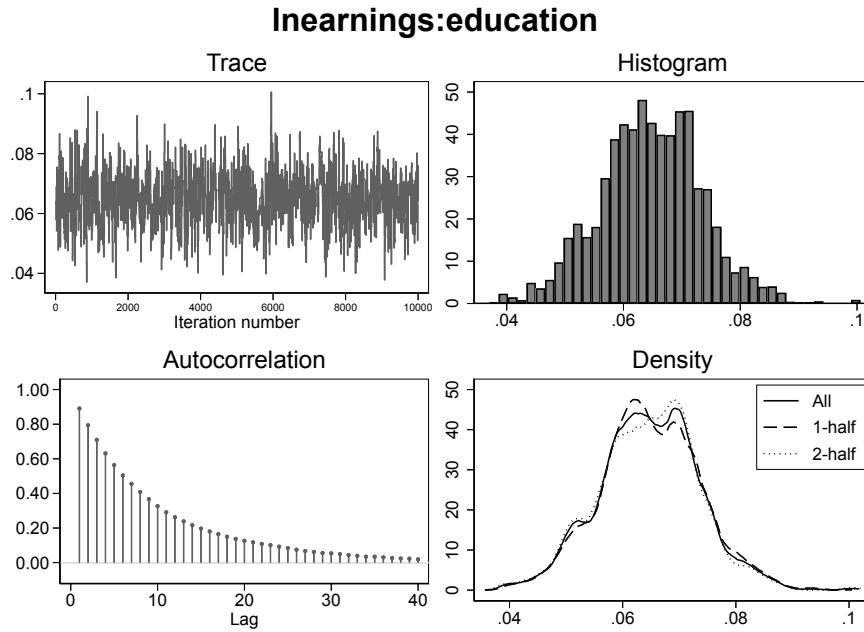
The column `ESS` gives the effective sample size. An `ESS` of 539.68 for  $\beta_{\text{ed}}$  means that the 10,000 correlated MH draws of  $\beta_{\text{ed}}$  are equivalent to 540 independent draws; see [BAYES] **bayesstats ess** for details on computation. The column `Efficiency` gives  $\text{ESS}/S$ , where  $S$  is the number of MH draws. The column `Corr. time` is the reciprocal of `Efficiency`. It is desirable to have efficiency greater than 0.10, but in practice efficiencies as low as 0.01 may be acceptable. Here the efficiency rates range from 0.05 to 0.08.

Command `bayesgraph` can be used to give the same four diagnostic graphs already presented in the i.i.d. case for each of the model parameters. The following command gives these graphs for the slope coefficient of variable `education`.

```

. * Diagnostic plots for MH posterior draws of beta_education
. bayesgraph diagnostics {lnearnings:education}

```



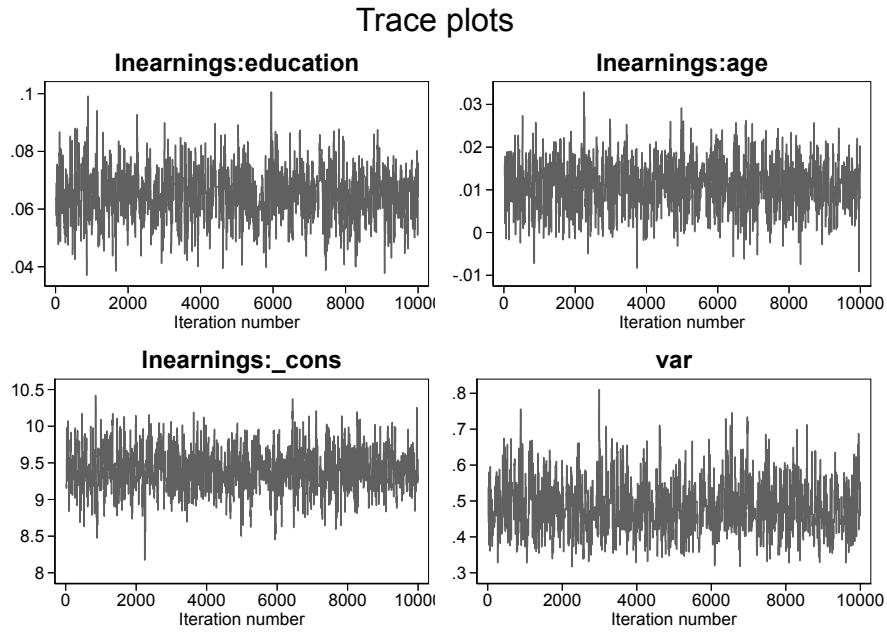
**Figure 29.4.** Diagnostics for posterior draws of education coefficient

The plots in figure 29.4 suggest that the chain has converged. However, the bottom left panel indicates considerable correlation in the draws, leading to low efficiency for the MH algorithm.

A specific diagnostic plot can be presented for several or all parameters. Here this is illustrated for the trace plot.

- . \* Trace plot for all four parameters
- . bayesgraph trace \_all, combine

The traces for all four parameters given in figure 29.4 seem reasonable.



**Figure 29.5.** Trace for all four coefficients

A very useful visual diagnostic is to compare the first and second half of all the MCMC draws for all parameters; an example is given in section [29.6.5](#).

#### 29.6.4 A supposed uninformative prior can be informative

It is easy to mistakenly specify a desired uninformative prior to be informative. A poor choice of prior for any single parameter, including nuisance parameters, can impact not only the posterior for that parameter but also the posterior for parameters of intrinsic interest.

Suppose we have no idea about where the intercept should be centered and specify an  $N(0, 1.0^2)$  prior for the intercept, along with priors of  $N(0.06, 0.1^2)$  and  $N(0.02, 0.1^2)$  for education and age. The prior for the intercept is seemingly less tight than that for the slope coefficients because the standard deviation is 10 times larger.

With this new prior, we obtain

```

. * Bayesian posterior with prior for intercept tighter and centered on zero
. qui bayesmh lnearnings education age, likelihood(normal({var}))
> rseed(10101) prior({lnearnings:education}, normal(0.06,0.01))
> prior({lnearnings:age}, normal(0.02,0.01))
> prior({lnearnings:_cons}, normal(0,1))
> prior({var}, igamma(1,0.5))

. bayesstats summary

```

Posterior summary statistics MCMC sample size = 10,000

|            | Mean     | Std. dev. | MCSE    | Median   | Equal-tailed<br>[95% cred. interval] |          |
|------------|----------|-----------|---------|----------|--------------------------------------|----------|
| lnearnings | .1495105 | .022584   | .001    | .1481057 | .1070548                             | .1944438 |
| education  | .026712  | .0070295  | .000329 | .0265557 | .0143138                             | .0402314 |
| age        | 7.510787 | .4769094  | .02173  | 7.518329 | 6.541839                             | 8.308205 |
| _cons      |          |           |         |          |                                      |          |
| var        | .5505413 | .0918929  | .005451 | .5368902 | .4038858                             | .7633164 |

The posterior mean for the intercept has changed somewhat, from 9.41 to 7.51, and is still a long way from the prior mean of 0. But the returns to education and to age have more than doubled from 0.065 to 0.149 and from 0.011 to 0.027!

This susceptibility of posterior results to specification of the prior cannot be overemphasized. When an informative prior is used, there should be a justification for the priors on all parameters, including seemingly innocuous parameters such as the intercept. Where there is little prior knowledge, the prior variance should be very large. And one should check the robustness of results to changes in the prior.

### 29.6.5 A prior can lead to an unidentified model being identified

Informative priors can aid in identification, though the source of identification may then be concealed.

To illustrate this, we add as regressor the regressor `educcopy`, which is a duplicate of variable `education`. The model is then no longer identified. OLS regression using command `regress` leads to automatic dropping of one of `education` or `educcopy`.

Now suppose we perform Bayesian analysis as in section [29.6.3](#), adding the additional regressor `educcopy` with  $N(0.10, 0.01^2)$  prior (and dropping `age` for brevity). We obtain

```
. * Bayesian posterior with same regressor appearing twice and informative prior
. generate educcopy = education
. qui bayesmh lnearnings education educcopy,
> rseed(10101) likelihood(normal({var}))
> prior({lnearnings:education}, normal(0.06,0.0001))
> prior({lnearnings:educcopy}, normal(0.10,0.0001))
> prior({lnearnings:_cons}, normal(10,100))
> prior({var}, igamma(1,0.5))
. bayesstats summary
```

Posterior summary statistics MCMC sample size = 10,000

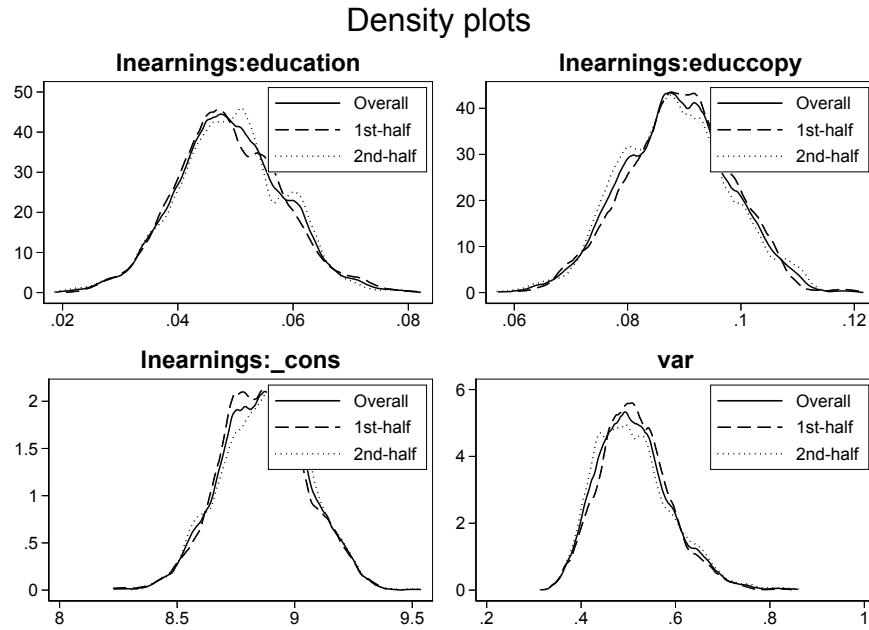
|            | Mean     | Std. dev. | MCSE    | Median   | Equal-tailed<br>[95% cred. interval] |          |
|------------|----------|-----------|---------|----------|--------------------------------------|----------|
| lnearnings |          |           |         |          |                                      |          |
| education  | .0490347 | .0092127  | .000324 | .0487151 | .0315036                             | .0674347 |
| educcopy   | .0886085 | .0093957  | .000406 | .0884671 | .0701305                             | .1071469 |
| _cons      | 8.870992 | .1811948  | .008026 | 8.871312 | 8.528059                             | 9.223388 |
| var        | .5123742 | .0778485  | .003532 | .5052402 | .3821599                             | .6845542 |

The variables `education` and `educcopy` are now both identified with posterior means of 0.04903 and 0.0886 and posterior standard deviations that are relatively small.

From output not given, the sampling efficiency for the three parameters ranged from 0.049 to 0.081, compared with the range 0.054 to 0.081 for the model of section [29.6.3](#).

As a check of chain convergence, we plot the density of the posterior for the first and second half of the draws for each parameter.

```
. * Posterior density for first and second half of draws for all parameters
. bayesgraph kdensity _all, show(both) combine
```



**Figure 29.6.** Density for each half of draws for all four coefficients

Figure 29.6 suggests that the chain has converged.

As a more rigorous test, we then run multiple chains, rerunning the previous `bayesmh` command with the additional option `nchains(5)`. The `bayesstats grubin` command then yields

```
. * Rc statistic from running five chains on preceding bayesmh command
. bayesstats grubin
```

Gelman-Rubin convergence diagnostic

```
Number of chains = 5
MCMC size, per chain = 10,000
Max Gelman-Rubin Rc = 2.391745
```

|            | Rc       |
|------------|----------|
| lnearnings |          |
| education  | 2.391745 |
| educcopy   | 2.379293 |
| _cons      | 1.002488 |
| var        | 1.002649 |

Convergence rule:  $R_c < 1.1$

The chain has clearly not converged, because the  $R_c$  statistics for `education` and `educcopy` are much greater than 1.1.

The Stata default is to use 2,500 burnin draws. Increasing the burn-in to 20,000 draws, adding option `burnin(20000)` to the previous `bayesmh` command, and running five chains lead to chain convergence because, from output not given, the  $R_c$  statistics for the four parameters then range from 1.0004 to 1.0012.

This example makes it clear that informative priors can lead to model identification. In some cases, this may be desired, but from section [29.6.4](#), it is easy to inadvertently specify an informative prior. A robustness test is to make the priors less informative. In the current example, increasing the variances of the priors for `education` and `educcopy` from 0.0001 to 1.0 leads to the two variables having posterior standard deviation more than 10 times the posterior mean, an indication of extreme multicollinearity.

## 29.7 Modifying the MH algorithm

Stata uses a random-walk adaptive MH algorithm. The proposal density  $q(\boldsymbol{\theta}^* | \boldsymbol{\theta}_{s-1})$  is the normal with mean  $\boldsymbol{\theta}_{s-1}$  and variance matrix  $\rho_k^2 \boldsymbol{\Sigma}_k$ , where  $\rho_k$  and  $\boldsymbol{\Sigma}_k$  are updated every 100 MH iterations. Various default settings for the adaptive MH algorithm can be changed using the options `adaptation()`, `scale()`, and `covariance()` of command `bayesmh`.

Here we consider improving the efficiency of the MH algorithm by applying the algorithm within blocks and by using the Gibbs sampler.

### 29.7.1 Blocking parameters

In its default form, the adaptive MH algorithm acts on the entire vector of parameters  $\boldsymbol{\theta}$ . Computational efficiency can improve if the parameters are broken into blocks, and the MH algorithm is applied recursively across the blocks. Blocks should be formed of parameters that are likely to be reasonably correlated with each other, while parameter correlation across blocks should be very low. The adaptations of the proposal densities are then done separately for each block.

A minimal blocking separates mean parameters  $\boldsymbol{\beta}$  from variance parameters  $\sigma^2$ .

```

. * MH with blocking: var in separate block
. qui bayesmh lnearnings education age, likelihood(normal({var}))
> prior({lnearnings:education}, normal(0.06,0.0001))
> prior({lnearnings:age}, normal(0.02,0.0001))
> prior({lnearnings:_cons}, normal(10,100))
> prior({var}, igamma(1,0.5)) block({var}) rseed(10101)
. bayesstats summary

```

Posterior summary statistics MCMC sample size = 10,000

|            | Mean     | Std. dev. | MCSE    | Median   | Equal-tailed<br>[95% cred. interval] |          |
|------------|----------|-----------|---------|----------|--------------------------------------|----------|
| lnearnings | .0647583 | .0092147  | .000311 | .0648271 | .047091                              | .0836823 |
| education  | .010574  | .0055013  | .00021  | .0105884 | -.0002566                            | .0213766 |
| age        | 9.415344 | .2905818  | .009854 | 9.398902 | 8.853972                             | 10.00916 |
| var        | .4785125 | .0701401  | .001664 | .4723414 | .3593001                             | .6328165 |

```
. bayesstats ess
```

Efficiency summaries MCMC sample size = 10,000  
 Efficiency: min = .06851  
 avg = .1052  
 max = .1776

|            | ESS     | Corr. time | Efficiency |
|------------|---------|------------|------------|
| lnearnings | 876.03  | 11.42      | 0.0876     |
| education  | 685.13  | 14.60      | 0.0685     |
| age        | 869.56  | 11.50      | 0.0870     |
| var        | 1776.22 | 5.63       | 0.1776     |

```
. di "Overall acceptance rate = " e(arate)
Overall acceptance rate = .3449655
```

The efficiency for  $\sigma^2$  has tripled, from 0.0557 to 0.1776, while the efficiencies for the slope parameters have increased somewhat.

## 29.7.2 Gibbs sampler within MH

The Gibbs sampler is an MCMC method that has the advantage of having a high acceptance rate (of one); see section [29.3.5](#). It can be used in Bayesian applications in cases where analytical results exist for one or more conditional posteriors.

For the current example with inverse-gamma prior for  $\sigma^2$ , an analytical result is available for the conditional posterior for  $\sigma^2$  given  $\beta$  (it is also inverse-gamma with known formulas for the shape and scale parameters).

Stata command `bayesmh` allows use of the Gibbs sampler for drawing  $\sigma_s^2$  given  $\beta_{s-1}$ , while the MH algorithm is used to draw  $\beta_s$  given  $\beta_{s-1}$  and  $\sigma_{s-1}^2$ . We obtain

```
. * Hybrid MH with Gibbs sampling subcomponent
. qui bayesmh lnearnings education age, likelihood(normal({var}))
> prior({lnearnings:education}, normal(0.06,0.0001))
> prior({lnearnings:age}, normal(0.02,0.0001))
> prior({lnearnings:_cons}, normal(10,100))
> prior({var}, igamma(1,0.5)) block({var}, gibbs) rseed(10101)
. bayesstats summary

Posterior summary statistics MCMC sample size = 10,000

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
lnearnings						
education	.0653215	.0090728	.000351	.065111	.0483262	.0836858
age	.010581	.0053998	.000174	.0106941	-.0001928	.0214931
_cons	9.411757	.2821473	.009121	9.417786	8.859742	9.956663
var	.4779449	.0686372	.000747	.4714431	.3615059	.6279229


```
. bayesstats ess

Efficiency summaries      MCMC sample size = 10,000
                           Efficiency: min = .06697
                                         avg = .2758
                                         max = .8444
```


	ESS	Corr. time	Efficiency
lnearnings			
education	669.72	14.93	0.0670
age	960.54	10.41	0.0961
_cons	956.93	10.45	0.0957
var	8444.33	1.18	0.8444


```
. di "Overall acceptance rate = " e(arate)
Overall acceptance rate = .66958304
```


```

Now the efficiency for  $\sigma^2$  is close to perfect, while there is some further improvement for the components of  $\beta$ .

## 29.8 RE model

The treatment of the regression model to date has assumed i.i.d. errors. In practice, errors are often grouped or clustered. We present Bayesian analysis of a RE model, with intercept that varies across groups, based on hierarchical priors. Richer multilevel models additionally allow slope coefficients to vary across groups.

### 29.8.1 RE MLE

We begin with ML estimation of the usual RE model for individual  $i$  in group  $j$ . Then,

$$y_{ij} = \beta_1 + \beta_2 x_{ij} + v_j + \varepsilon_{ij}$$

where  $v_j \sim \mathcal{N}[0, \sigma_v^2]$  and  $\varepsilon_{ij} \sim \mathcal{N}[0, \sigma_\varepsilon^2]$ .

The ML estimates of  $\beta_1, \beta_2, \sigma_v^2$ , and  $\sigma_\varepsilon^2$  can be obtained using command `mixed`.

As an example, we adapt the model in section 29.6.3 to have a separate random intercept at each level of `education` rather than entering `education` as a continuous regressor. Thus, we regress `lnearnings` ( $y_{ij}$ ) on an intercept and `age` ( $x_{ij}$ ) with grouping ( $j$ ) on each discrete value of `education`. We obtain

```

. * Mixed-effects estimation of RE model
. qui use mus229acs, clear
. quietly keep if _n <= 100
. mixed learnings age || education: , nolog

Mixed-effects ML regression
Group variable: education
Number of obs = 100
Number of groups = 11
Obs per group:
min = 1
avg = 9.1
max = 33
Wald chi2(1) = 1.20
Prob > chi2 = 0.2730
Log likelihood = -107.28605

```

| llearnings | Coefficient | Std. err. | z     | P> z  | [95% conf. interval] |
|------------|-------------|-----------|-------|-------|----------------------|
| age        | .0070023    | .0063881  | 1.10  | 0.273 | -.0055181 .0195228   |
| _cons      | 10.35348    | .3266064  | 31.70 | 0.000 | 9.713341 10.99362    |

| Random-effects parameters         | Estimate | Std. err. | [95% conf. interval] |
|-----------------------------------|----------|-----------|----------------------|
| education: Identity<br>var(_cons) | .1566638 | .1808768  | .0163009 1.505659    |
| var(Residual)                     | .4457641 | .0708718  | .3264172 .6087474    |

LR test vs. linear model: chibar2(01) = 4.31                          Prob >= chibar2 = 0.0189

The estimated intercept and slope are, respectively, 10.35 and 0.0070, and the estimated error variances are  $\hat{\sigma}_v^2 = 0.157$  and  $\hat{\sigma}_\epsilon^2 = 0.446$ .

For comparison with subsequent analysis, we additionally calculate the best linear unbiased predictors of the group-specific errors  $v_j$  and the corresponding group-specific intercepts  $\beta_{1j} = \beta_1 + v_j$  at each level of education.

```

. * Predict the random intercepts = intercept + RE
. predict u0, reffects
. by education, sort: generate tolist = (_n==1)
. generate randomint = _b[_cons] + u0
. list education u0 randomint if tolist, clean

 educat~n u0 random~t
1. 0 -.5186173 9.834861
3. 6 -.1734726 10.18001
4. 8 -.1500419 10.20344
5. 9 -.1516948 10.20178
6. 10 .4270733 10.78055
7. 12 .0500333 10.40351
40. 13 -.0934182 10.26006
56. 14 -.2972531 10.05622
62. 16 .3830092 10.73649
91. 18 .1542939 10.50777
99. 20 .3700881 10.72357

```

The predicted intercepts range from 9.83 to 10.78 and are highest for completed degrees (at 10, 12, 16, 18, and 20 years of education).

## 29.8.2 Bayesian RE models using hierarchical priors

The preceding RE model can equivalently be written as a model with intercept that varies with group  $j$ . The model is

$$y_{ij} = \beta_{1j} + \beta_2 x_{ij} + \varepsilon_{ij}$$

where  $\beta_{1j} \sim N(\beta_1, \sigma_v^2)$  and  $\varepsilon_{ij} \sim N(0, \sigma_\varepsilon^2)$ .

A hierarchical prior for the intercept first specifies a prior for  $\beta_{1j}$ , here the  $N(\beta_1, \sigma_v^2)$  distribution. Then, priors are specified for the parameters  $\beta_1$  and  $\sigma_v^2$  of the original prior for  $\beta_{1j}$ .

We again consider regression of `lnearnings` on an intercept and `age`, where the intercept varies by level of `education`.

A simple approach is to use default priors and apply the `bayes` prefix to the `mixed` command used in section [29.8.1](#).

```
. bayes, showreffects rseed(10101): mixed lnearnings age || education:
(output omitted)
```

The `showreffects` option leads to output that includes the random effects.

The `bayes: mixed` command uses Gibbs sampling for regression coefficients and variance components and uses adaptive MH for the random effects. The command is restricted to normal priors for random effects and does not provide an ability to change this.

The `bayesmh` command with the multilevel specification introduced in Stata 17 provides much more flexibility. The random effects with grouping on level of `education` can be specified as `v[education]`. Then, a variance component `var_v` is created, and `v[education]` has an  $N(0, var_v)$  prior. Other capitalized letters such as `U` could instead be used. For illustrative purposes, we also specify the priors, rather than use defaults. The hierarchical priors are specified to be  $N(10, 10^2)$  for `_cons` and inverse-gamma (0.001, 0.001) for `var_0`. In this example, the Gibbs sampler can be used throughout, so we use the `block(..., gibbs split)` option with all blocks of parameters (regression coefficients, variance components, and random effects) included. We obtain

```

. * Bayesian hierarchical model using bayesmh with multilevel specification
. bayesmh lnearnings age V[education], rseed(10101)
> likelihood(normal({var_0})) showreffects
> prior({lnearnings:_cons}, normal(10,100))
> prior({lnearnings:age}, normal(0,0.01))
> prior({var_0}, igamma(0.0001, 0.0001))
> prior({var_V}, igamma(0.0001, 0.0001))
> block({lnearnings:} {var_0} {var_V} {V}, gibbs split)
Burn-in 2500 aaaaaaaaaaa1000aaaaaaaa2000aaaaaa done
Simulation 100001000.....2000.....3000.....
> 4000.....5000.....6000.....7000.....
> 8000.....9000.....10000 done

```

---

Model summary

Likelihood:

$\text{lnearnings} \sim \text{normal}(\text{xb\_lnearnings}, \{\text{var}_0\})$

Priors:

|                    |                         |     |
|--------------------|-------------------------|-----|
| {lnearnings:_cons} | ~ normal(10,100)        | (1) |
| {lnearnings:age}   | ~ normal(0,0.01)        | (1) |
| {V[education]}     | ~ normal(0,{var_V})     | (1) |
| {var_0}            | ~ igamma(0.0001,0.0001) |     |

Hyperprior:

$\{\text{var}_V\} \sim \text{igamma}(0.0001,0.0001)$

---

(1) Parameters are elements of the linear form  $\text{xb\_lnearnings}$ .

|                            |                    |        |
|----------------------------|--------------------|--------|
| Bayesian normal regression | MCMC iterations =  | 12,500 |
| Gibbs sampling             | Burn-in =          | 2,500  |
|                            | MCMC sample size = | 10,000 |
|                            | Number of obs =    | 100    |
|                            | Acceptance rate =  | 1      |
|                            | Efficiency: min =  | .01551 |
|                            | avg =              | .1382  |
| Log marginal-likelihood    | max =              | .4035  |

|              | Mean      | Std. dev. | MCSE    | Median    | Equal-tailed<br>[95% cred. interval] |          |
|--------------|-----------|-----------|---------|-----------|--------------------------------------|----------|
| lnearnings   |           |           |         |           |                                      |          |
| age          | .0064714  | .0070284  | .000528 | .0062262  | -.006898                             | .0204813 |
| _cons        | 10.38439  | .3732316  | .029968 | 10.39792  | 9.60542                              | 11.0615  |
| var_0        | .4700626  | .07673    | .001575 | .4615064  | .3426978                             | .6421693 |
| var_V        | .2488358  | .2868117  | .011141 | .16028    | .0019257                             | 1.014185 |
| V[education] |           |           |         |           |                                      |          |
| 0            | -.5035241 | .4451703  | .015694 | -.438133  | -1.521537                            | .1366138 |
| 6            | -.186511  | .3900403  | .007276 | -.124512  | -1.08712                             | .5032018 |
| 8            | -.1568481 | .3800651  | .005983 | -.1008047 | -1.04303                             | .5179899 |
| 9            | -.1586843 | .3846119  | .006312 | -.1023517 | -1.039526                            | .5410226 |
| 10           | .4690828  | .503758   | .020851 | .3648499  | -.2273046                            | 1.664463 |
| 12           | .0466093  | .2034901  | .007634 | .0305266  | -.342077                             | .4904817 |
| 13           | -.0795536 | .2194784  | .006561 | -.0698219 | -.5219822                            | .3719066 |
| 14           | -.2596706 | .2745591  | .007767 | -.2334558 | -.849786                             | .2167593 |
| 16           | .346646   | .2346539  | .013663 | .3302939  | -.0315136                            | .8540507 |
| 18           | .1459382  | .254649   | .008329 | .114742   | -.3120736                            | .7104607 |
| 20           | .3715104  | .3948125  | .012248 | .3129923  | -.2345209                            | 1.267298 |

The posterior means of the intercept parameter and the `age` slope parameter are, respectively, 10.38 and 0.0065, compared with ML estimates of 10.35 and 0.0070 from command `mixed`. And the posterior means of the model variances are 0.249 and 0.470 compared with ML estimates of 0.157 and 0.446. The corresponding posterior standard deviations are within 20% of the ML standard errors, aside from the variance of the random intercept (`var_V`).

The average sampling efficiency of 0.1382 is good. Note that RE models can introduce many more parameters (here 12), leading to computational challenges so that greater care is needed to confirm that the chain is likely to have converged. The acceptance rate is 1 because in this special case, all conditional distributions were available, so only the Gibbs sampler is used.

The preceding command can be adapted to specify alternative prior distributions. For example, the option `prior({lnearnings:_cons}, t(10,100,5))` specifies a Student's  $t$  prior for the intercept with mean 10, squared scale 100, and 5 degrees of freedom. In that case, the default random-walk MH sampling needs to be used because Gibbs sampling is no longer possible.

## 29.9 Bayesian model selection

The principal tools for Bayesian model selection are Bayes factors and the posterior odds ratio. These can be applied to either nested or nonnested models.

### 29.9.1 Bayes factors

Bayes factors are based on the marginal likelihood

$$m(\mathbf{y}|\mathbf{X}) = \int L(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X}) \times \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}.$$

Let  $m_1(\mathbf{y}|\mathbf{X})$  and  $m_2(\mathbf{y}|\mathbf{X})$  denote the marginal likelihoods of models 1 and 2. Then the Bayes factor is defined as

$$\text{Bayes factor} = B_{12} = \frac{m_1(\mathbf{y}|\mathbf{X})}{m_2(\mathbf{y}|\mathbf{X})}$$

Model 1 is preferred to model 2 if  $B_{12} > 1$ . A commonly used rule of thumb is that evidence against model 2 is weak if  $1 < B_{12} < 3$ , substantial if  $3 < B_{12} < 20$ , strong if  $20 < B_{12} < 150$ , and very strong if  $B_{12} > 150$ . This rule of thumb does not account for the relative size of the two models.

However, the marginal likelihood  $m(\mathbf{y}|\mathbf{X})$  is difficult to compute—recall that MCMC methods could obtain draws from the posterior without computing the marginal likelihood. Several methods have been proposed to numerically estimate the marginal likelihood; see [Kass and Raftery \(1995\)](#) and references in [Cameron and Trivedi \(2005\)](#), 457–458). Stata command `bayesmh` uses the Laplace–Metropolis estimator

$$\hat{m} = (2\pi)^{p/2} |\widehat{\mathbf{S}}|^{1/2} p\left(\mathbf{y}|\widetilde{\boldsymbol{\theta}}\right) \pi\left(\widetilde{\boldsymbol{\theta}}\right)$$

where  $\hat{S}$  is the posterior sample variance and  $\tilde{\theta}$  is the mode of the posterior density. This quantity  $\hat{m}$  is included in the output from command `bayesmh`. It can always be computed given MCMC draws from the posterior, even when the posterior is improper. Considerable care is therefore needed. In particular, [Koop \(2003, 42\)](#) states that noninformative priors should be used only for parameters common to both models; otherwise, informative priors should be used.

As an example, we compare two nested models, one with both `education` and `age` as regressors and one with just `age` as a regressor. The same priors are used for the parameters common to both models. The results for the larger model were already saved earlier. For the smaller model, we have

```
. * Bayesian posterior for small model without regressor education
. qui bayesmh lnearnings age, likelihood(normal({var}))
> prior({lnearnings:age}, normal(0.02,0.0001))
> prior({lnearnings:_cons}, normal(10,100))
> prior({var}, igamma(1,0.5))
> saving(mcmcdraws_smallregress, replace) rseed(10101)
. estimates store smallregress
. di "Marginal likelihood for smaller model " e(lml_lm)
Marginal likelihood for smaller model -117.77778
```

The Bayes factor can be computed using command `bayesstats ic`. Here we compare just two models, but several models can be listed. The default is for the base model to be the first listed model. Here we use option `bayesfactor` to have the Bayes factor listed; the default is to instead print the natural logarithm of the Bayes factor.

```
. * Bayes factor with first-listed model the base model
. bayesstats ic fullregress smallregress, bayesfactor
Bayesian information criteria
```

|              | DIC      | log(ML)   | BF       |
|--------------|----------|-----------|----------|
| fullregress  | 211.5527 | -111.2892 | .        |
| smallregress | 225.06   | -117.7778 | .0015207 |

Note: Marginal likelihood (ML) is computed using Laplace-Metropolis approximation.

The Bayes factor is much less than one, very strongly favoring the base model, which includes both `education` and `age` as regressors. Note that  $B_{21} = m_2/m_1 = e^{-117.78}/e^{-111.29} = 0.00152 < 1/150$ . The output also includes the deviance information criterion (DIC), which is based on the deviance statistic for the likelihood; see [BAYES] **bayesstats ic** for details. Models with smaller values of DIC are preferred, so the first model is preferred. In large samples, it can be shown that one half the difference in DIC across two models equals the natural logarithm of the Bayes factor. Here  $\Delta \text{DIC}/2 = -6.75$  and  $-\ln(0.0015207) = -6.49$ .

## 29.9.2 Posterior odds ratio

Additionally, one may have prior probabilities  $p_1$  and  $p_2 = 1 - p_1$  for models 1 and 2. In that case, the Bayes factor is weighted by the ratio of these prior probabilities to give the posterior odds ratio

$$\text{Posterior odds} = B_{12} \times \frac{p_1}{p_2} = \frac{m_1(\mathbf{y}|\mathbf{X})}{m_2(\mathbf{y}|\mathbf{X})} \times \frac{p_1}{p_2}$$

Command `bayesstats model` can be used to compute the posterior odds ratio. In this example, we place prior odds of 0.8 on the larger model and 0.2 on the smaller model. The larger model is clearly favored because it has  $P(M|y)$  much greater than 0.5.

```
. * Posterior odds Bayes factor with first-listed model the base model
. bayestest model fullregress smallregress, prior(0.8 0.2)
```

Bayesian model tests

|              | $\log(\text{ML})$ | $P(M)$ | $P(M y)$ |
|--------------|-------------------|--------|----------|
| fullregress  | -111.2892         | 0.8000 | 0.9996   |
| smallregress | -117.7778         | 0.2000 | 0.0004   |

Note: Marginal likelihood (ML) is computed using Laplace–Metropolis approximation.

## 29.10 Bayesian prediction

Most prediction methods give a single prediction for an observation. Bayesian methods can be used to obtain a posterior predictive distribution that provides a distribution, not just a single prediction.

### 29.10.1 Posterior predictive distribution

Bayesian analysis treats both  $\mathbf{y}$  and knowledge of  $\boldsymbol{\theta}$  as random. Given a specified likelihood function  $L(\mathbf{y}|\boldsymbol{\theta})$  and prior  $\pi(\boldsymbol{\theta})$ , the marginal density of  $\mathbf{y}$  or marginal likelihood is then  $p(\mathbf{y}) = \int p(\mathbf{y}, \boldsymbol{\theta}) d\boldsymbol{\theta} = \int L(\mathbf{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}$ .

Suppose we wish to predict new observations  $\mathbf{y}^{\text{new}}$  conditioning on existing observations  $\mathbf{y}^{\text{obs}}$ . The posterior predictive distribution of  $\mathbf{y}^{\text{new}}$  given  $\mathbf{y}^{\text{obs}}$  is obtained as

$$\begin{aligned} p(\mathbf{y}^{\text{new}}|\mathbf{y}^{\text{obs}}) &= \int p(\mathbf{y}^{\text{new}}, \boldsymbol{\theta}|\mathbf{y}^{\text{obs}}) d\boldsymbol{\theta} \\ &= \int p(\mathbf{y}^{\text{new}}|\boldsymbol{\theta}, \mathbf{y}^{\text{obs}}) p(\boldsymbol{\theta}|\mathbf{y}^{\text{obs}}) d\boldsymbol{\theta} \\ &= \int L(\mathbf{y}^{\text{new}}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathbf{y}^{\text{obs}}) d\boldsymbol{\theta} \end{aligned}$$

where the last equality assumes independence between  $\mathbf{y}^{\text{new}}$  and  $\mathbf{y}^{\text{obs}}$  conditional on  $\boldsymbol{\theta}$ .

Here  $L(\mathbf{y}^{\text{new}}|\boldsymbol{\theta})$  is the likelihood for the new observations, and  $p(\boldsymbol{\theta}|\mathbf{y}^{\text{obs}})$  is the posterior. By making  $B$  MCMC draws from the posterior  $p(\boldsymbol{\theta}|\mathbf{y}^{\text{obs}})$  and computing  $L(\mathbf{y}^{\text{new}}|\boldsymbol{\theta})$  at each of these draws, we obtain  $B$  draws from the posterior predictive distribution of  $\mathbf{y}^{\text{new}}$  given  $\mathbf{y}^{\text{obs}}$ .

For regression, the posterior predictive distribution is

$$p(\mathbf{y}^{\text{new}}|\mathbf{y}^{\text{obs}}, \mathbf{X}^{\text{new}}) = \int L(\mathbf{y}^{\text{new}}|\boldsymbol{\theta}, \mathbf{X}^{\text{new}}) p(\boldsymbol{\theta}|\mathbf{y}^{\text{obs}}, \mathbf{X}^{\text{obs}}) d\boldsymbol{\theta}$$

and MCMC draws are obtained similarly.

### 29.10.2 The bayespredict command

The Bayesian prediction commands `bayespredict` and `bayesstats ppvalues` are available only after `bayesmh`; they are not available after the simpler `bayes` prefix.

The `bayespredict` postestimation command obtains many MCMC draws for each observation and stores them in a file,

```
bayespredict ysimspec [ysimspec ...] [if] [in], saving(filespec) [simopts]
```

where in the simplest case *ysimspec* is `{_ysim}`. More generally, one can specify a function to be predicted, in which case *funcspec* replaces *ysimspec*. The default is to obtain 10,000 draws.

Often, one is interested in only a summary of the draws, such as the posterior mean for each observation. This can be obtained using the command

```
bayespredict [type] newvarspec [if] [in], mean
```

To obtain the posterior median, standard deviation, and credible interval, replace `mean` with, respectively, `median`, `std`, and `cri`.

### 29.10.3 Bayesian prediction example

Bayesian prediction has many purposes, including multiple imputation (see section [30.5](#)) and model checking.

Here we consider a simple example where predictions of log earnings are made for 2 individuals, 1 with 12 years of education and aged 40 years and 1 with 16 years of education and aged 40 years.

We fit the same model as in section [29.6.3](#) based on 100 observations.

```

. * Bayesian posterior with informative priors: Normal for b, inv gamma for s2
. qui bayesmh lnearnings education age, likelihood(normal({var}))
> rseed(10101) prior({lnearnings:education}, normal(0.06,0.0001))
> prior({lnearnings:age}, normal(0.02,0.0001))
> prior({lnearnings:_cons}, normal(10,100))
> prior({var}, igamma(1,0.5)) saving(mcmcdraws_fullregress, replace)

```

The explanatory variables for the 2 new observations are generated as follows, using knowledge that the original sample had 100 observations.

```

. * Create 2 new observations (numbers 101 and 102) for prediction
. set obs 102
Number of observations (_N) was 100, now 102.
. quietly replace education = 12 if _n == 101
. quietly replace age = 40 if _n == 101
. quietly replace education = 16 if _n == 102
. quietly replace age = 40 if _n == 102

```

The following command yields 10,000 MCMC draws from the posterior predictive density for each of the 2 individuals.

```

. * Obtain 10,000 MCMC predictions for each of the 2 individuals
. bayespredict {_ysim1} if _n > 100, saving(mcmcpredict, replace) rseed(10101)
Computing predictions ...
file mcmcpredict.dta not found; file saved.
file mcmcpredict.ster not found; file saved.

```

For the first individual, the 10,000 predictions of the actual value of log-earnings have mean 10.608, while the 10,000 predictions of the mean ( $\mathbf{x}'\beta$ ) have average 10.613.

```

. * Summarize the MCMC predictions dataset
. preserve
. use mcmcelpred, clear
. summarize

```

| Variable   | Obs    | Mean     | Std. dev. | Min      | Max      |
|------------|--------|----------|-----------|----------|----------|
| _chain     | 10,000 | 1        | 0         | 1        | 1        |
| _index     | 10,000 | 5000.5   | 2886.896  | 1        | 10000    |
| _ysim1_101 | 10,000 | 10.60773 | .7054899  | 7.29318  | 13.20271 |
| _ysim1_102 | 10,000 | 10.87181 | .6986186  | 7.716644 | 13.52072 |
| _mu1_101   | 10,000 | 10.61325 | .0728873  | 10.33691 | 10.86419 |
| <hr/>      |        |          |           |          |          |
| _mu1_102   | 10,000 | 10.87211 | .0726094  | 10.61295 | 11.13556 |
| _frequency | 10,000 |          | 1         | 0        | 1        |
| <hr/>      |        |          |           |          |          |
| . restore  |        |          |           |          |          |

The bayesstats summary command can be used to provide summaries of the posterior predictive distribution for the two observations.

| Posterior summary statistics |          |           |         |          |                      | MCMC sample size = 10,000 |
|------------------------------|----------|-----------|---------|----------|----------------------|---------------------------|
|                              | Mean     | Std. dev. | MCSE    | Median   | [95% cred. interval] | Equal-tailed              |
| _ysim1_101                   | 10.60773 | .7054899  | .007055 | 10.60946 | 9.208361             | 11.99379                  |
| _ysim1_102                   | 10.87181 | .6986186  | .006986 | 10.87383 | 9.501282             | 12.2324                   |

Note that the 95% credible regions are very broad because they are for forecasts of the actual value of log earnings rather than for prediction of the conditional mean of log earnings; see section 4.2.5 for discussion of this important distinction.

There is no need to first generate and save a file with the 10,000 MCMC draws if interest lies only in summary statistics such as the mean of the posterior predictive distribution. For example,

```
. * Directly obtain the mean of the posterior predictive distribution.
. bayespredict plnearnings if _n > 100, mean rseed(10101)
Computing predictions ...
. list plnearnings if _n > 100, clean
 plnear^s
101. 10.60773
102. 10.87181
```

## 29.11 Probit example

The same MH algorithm can be applied to nonlinear regression models such as the probit model.

### 29.11.1 MLE

We continue with the same dataset but consider probit regression where the dependent variable equals 1 if annual earnings exceed \$75,000 and equals 0 otherwise.

```
. * Create dependent variable for probit example
. qui use mus229acs, clear
. quietly keep if _n <= 100
. generate dhighearns = earnings > 75000
. quietly save musrearnings, replace
. summarize dearings age education
```

| Variable  | Obs | Mean  | Std. dev. | Min | Max |
|-----------|-----|-------|-----------|-----|-----|
| dearnings | 100 | .2    | .4020151  | 0   | 1   |
| age       | 100 | 43.33 | 10.9342   | 25  | 65  |
| education | 100 | 13.69 | 3.158106  | 0   | 20  |

Here 20% of the sample has high earnings.

We regress dhighearns on an intercept, education and age by ML using command `probit`.

```
. * MLE for the probit regression
. probit dhighearns education age, nolog
Probit regression
Number of obs = 100
LR chi2(2) = 6.66
Prob > chi2 = 0.0358
Pseudo R2 = 0.0604
Log likelihood = -51.778422
```

| dhighearns | Coefficient | Std. err. | z     | P> z  | [95% conf. interval] |
|------------|-------------|-----------|-------|-------|----------------------|
| education  | .1184497    | .055289   | 2.14  | 0.032 | .0100852 .2268142    |
| age        | .0199199    | .0138726  | 1.44  | 0.151 | -.0072699 .0471097   |
| _cons      | -3.243121   | 1.071253  | -3.03 | 0.002 | -5.342738 -1.143505  |

The slope coefficients are 0.118 and 0.020. From output given at the end of section [29.11.3](#), the corresponding average marginal effects (AMES), obtained using command `margins`, are 0.035 and 0.006. So, for example, one more year of education is associated with a 0.035 increase in the probability of earnings exceeding \$75,000.

## 29.11.2 Bayesian analysis

We next perform Bayesian analysis of the same model.

The simplest approach is to use the `bayes` prefix with default normal uninformative priors, here `bayes: probit dhighearns education age`.

Instead, we use the `bayesmh` command with flat priors. This gives essentially the same results. We obtain

```
. * Bayesian posterior for probit regression with flat priors for beta
. bayesmh dhighearns education age, rseed(10101) likelihood(probit)
> prior({dhighearns:}, flat) saving(mcmcdraws_probit, replace)
Burn-in ...
Simulation ...
Model summary
```

---

Likelihood:  
  `dhighearns ~ probit(xb_dhighearns)`

Prior:  
  `{dhighearns:education age _cons} ~ 1 (flat)` (1)

---

(1) Parameters are elements of the linear form `xb_dhighearns`.

```

Bayesian probit regression
Random-walk Metropolis-Hastings sampling
Log marginal-likelihood = -58.189797
 MCMC iterations = 12,500
 Burn-in = 2,500
 MCMC sample size = 10,000
 Number of obs = 100
 Acceptance rate = .2726
 Efficiency: min = .08428
 avg = .08626
 max = .08879

```

| dhighearns | Equal-tailed |           |         |          |                      |           |
|------------|--------------|-----------|---------|----------|----------------------|-----------|
|            | Mean         | Std. dev. | MCSE    | Median   | [95% cred. interval] |           |
| education  | .128553      | .0549554  | .001844 | .1268172 | .0218047             | .2425331  |
| age        | .0212355     | .013818   | .000476 | .0212071 | -.0049327            | .0497453  |
| _cons      | -3.45482     | 1.088155  | .037168 | -3.43342 | -5.695359            | -1.481793 |

```
file mcmcdraws_probit.dta not found; file saved.
```

The posterior means for `education` and `age` are, respectively, 0.1286 and 0.0212, similar to the ML estimates of 0.1185 and 0.0199. The posterior standard deviations for `education` and `age` are, respectively, 0.0550 and 0.0138, very close to the ML standard errors of 0.0553 and 0.0139.

This result is expected. Because a noninformative prior is used, the Bayesian posterior means and standard deviations will be very close to the ML parameter estimates and standard errors. The Bayesian estimates, however, can be given a Bayesian interpretation.

### 29.11.3 MEs

From section 10.4.6, the probit coefficients can be directly interpreted up to a ratio because the probit model is a single index model. For example, the ME of one more year of education is roughly equivalent to that from 6 years of aging because the posterior mean of `education` is 6.05( $= 0.1286/0.0212$ ) times that of `age`. More precisely, we should compute the posterior mean of  $\beta_{\text{ed}}/\beta_{\text{age}}$  using command `bayesstats summary`.

But this still leaves the question of the extent to which the probability of having high earnings changes. To obtain MES requires additional coding after command `bayesmh`.

The AME of a change in the  $j$ th regressor evaluated at the  $s$ th posterior draw  $\beta_s$  is

$$\text{AME}_{js} = \frac{1}{N} \sum_{i=1}^N \phi(\mathbf{x}'_i \beta_s) \times \beta_{js}$$

where  $\phi(\cdot)$  is the standard normal density function.

The AMES can be calculated by applying the `bayespredict` command to a user-defined program that computes the AMES.

We first define a program we label `ameprog`.

```
. * Program to compute AME = phi(x`b)*b_j to be used by bayespredict
. program ameprog
1. version 17.0
2. args sum mu
3. // Define locals with shorter names for convenience
. local touse $BAYESPR_touse
4. local theta $BAYESPR_theta
5. local j $BAYESPR_passthruopts
6. // Obtain the current MCMC value of the jth coefficient
. tempname betaj
7. scalar `betaj' = `theta'[1, `j']
8. // Compute phi(xb) and store it in temporary variable tmpv
. tempvar tmpv
9. generate double `tmpv' = normalden(invnormal(`mu')) if `touse'
10. summarize `tmpv' if `touse', meanonly
11. // Store the final result, AME, in temporary scalar sum
. scalar `sum' = r(mean)*`betaj'
12. end
```

The program has two arguments: `sum` and `mu`. `sum` contains the name of a temporary scalar that stores the final result, here the AME. `mu` is a temporary variable that contains the linear prediction and is automatically filled in by `bayespredict` and passed to the program. Special global macros, `$BAYESPR*`, contain additional information that `bayespredict` passes to the program. In our example, we need the “touse” variable (`$BAYESPR_touse`), which marks the observations to be used in the computation; the MCMC coefficient values (`$BAYESPR_theta`), which are provided by `bayespredict`

in a temporary vector (one MCMC value at a time); and the additional options passed to the command and stored in global \$BAYESPR\_passthruopts. In our example, the additional option is the position of the coefficient for which we want to compute the AME.

Next, we call `bayespredict` to compute AMEs for the three coefficients by using our own `ameprog` program; `{_mu1}` refers to the linear predictor

```
. * Program that computes AMEs in each of 10,000 (the default) MCMC draws
. bayespredict (ame1:@ameprog {_mu1}, passthruopts(1))
> (ame2:@ameprog {_mu1}, passthruopts(2))
> (ame3:@ameprog {_mu1}, passthruopts(3)), saving(amepred, replace)
Computing predictions ...
file amepred.dta not found; file saved.
file amepred.ster not found; file saved.
```

Finally, we use `bayesstats summary` to compute posterior summaries of the 10,000 AMEs for the three regressors `education`, `age`, and the intercept.

```
. * Summary statistics for the AMEs from 10,000 MCMC draws
. bayesstats summary {ame1} {ame2} {ame3} using amepred
Posterior summary statistics MCMC sample size = 10,000

```

|      | Mean      | Std. dev. | MCSE    | Median    | Equal-tailed<br>[95% cred. interval] |           |
|------|-----------|-----------|---------|-----------|--------------------------------------|-----------|
| ame1 | .0361042  | .0146516  | .000503 | .035981   | .0062222                             | .0646957  |
| ame2 | .0059395  | .0037616  | .000131 | .0059721  | -.0013581                            | .013301   |
| ame3 | -.9684725 | .2683455  | .009798 | -.9728186 | -1.471454                            | -.4514302 |

The AME for `education` has posterior mean 0.0361, posterior standard deviation 0.0147 and 95% credible region [0.0062, 0.0647].

The Bayesian AMEs are very similar to those obtained using classical methods after ML estimation, as expected given the use of flat priors in this example.

```
. * Compare Bayesian AME to AME from ML estimation
. qui probit dhighearns education age
. margins, dydx(*)
```

Average marginal effects  
Model VCE: OIM

Number of obs = 100

Expression: Pr(dhighearns), predict()  
dy/dx wrt: education age

|           | Delta-method |           |      |       |                      |          |
|-----------|--------------|-----------|------|-------|----------------------|----------|
|           | dy/dx        | std. err. | z    | P> z  | [95% conf. interval] |          |
| education | .0345357     | .0152421  | 2.27 | 0.023 | .0046617             | .0644097 |
| age       | .0058079     | .0039359  | 1.48 | 0.140 | -.0019063            | .0135222 |

## 29.12 Additional resources

The [BAYES] *Stata Bayesian Analysis Reference Manual* provides a very detailed exposition of Bayesian analysis and MCMC computation. The key command is `bayesmh`. Bayesian analysis is presented in many books, including [Koop \(2003\)](#), [Gelman et al. \(2013\)](#), and [Cameron and Trivedi \(2005, chap. 13\)](#).

## 29.13 Exercises

1. Consider the log-earnings example introduced in section [29.2](#) and analyzed further in later sections. Give command `keep if _n > 600 & gender==1` to restrict analysis to a subset of women. Perform analysis with default priors using `bayes, rseed(10101): regress lnearnings education age hours`. Note that `hours` is also a regressor. Do the statistics given in the output suggest any convergence problems? Find the probability that the coefficient of `education` exceeds 0.10. Compare the Bayesian estimates with ML estimates and comment.
2. Continuing with the previous example, give command `bayesgraph diagnostics` separately for each of the four regressors and for the error variance (`sigma2`). And give command `bayesgraph ac _all, combine`. Finally, repeat the `bayes` command of question 1, adding the option `nchains(5)`, and then give command `bayesstats grubin`. Do you see any problems? Explain.
3. Continue with the same example as in questions 1 and 2, but use the `bayesmh` command, and specify the same informative prior as at the start of section [29.6.3](#), adding an  $N(0.2, 0.1^2)$  prior for the additional regressor `hours`. Compare estimates with the priors and with the estimates in question 1.
4. Compare, on the basis of the Bayes factor, the model of question 1 with a smaller model that omits `age` and `hours`. Which model do you prefer?
5. Suppose  $y$  is exponential distributed with density  $f(y) = \theta e^{-\theta y}$  and mean  $E(y) = 1/\theta$ . Let  $\mathbf{y} = (y_1, \dots, y_N)$  be data from a random sample of size  $N$ . Show that the likelihood is  $L(\mathbf{y}|\theta) = \theta^N e^{-\theta N \bar{y}}$ , where  $\bar{y}$  is the sample mean. Show that the MLE is  $\hat{\theta} = 1/\bar{y}$ . Suppose the prior for  $\theta$  is exponential with density  $\pi(\theta) = 2e^{-2\theta}$ , so the prior has mean  $E(\theta) = 1/2$ . Show that the resulting posterior density  $p(\theta|\mathbf{y}) \propto \theta^N e^{-\theta(N\bar{y}+1)}$ . One way to write a gamma density with parameters  $a$  and  $b$  is  $f(x) = \{b^{-a}/\Gamma(a)\}x^{a-1}e^{-x/b}$ , where  $\Gamma(\cdot)$  is the gamma function, the mean  $E(x) = ab$ , and variance  $\text{Var}(x) = ab^2$ . Given this information, obtain the mean and variance of the posterior density for  $\theta$  as a function of  $N$  and  $\bar{y}$ .
6. Consider Bayesian analysis of the model  $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}; \mathbf{u} \sim N(\mathbf{0}, \mathbf{I})$  with prior  $\boldsymbol{\beta} \sim N(\mathbf{0}, c \times \mathbf{I})$ , where the constant  $c$  is varied below and

for simplicity  $\sigma^2 = 1$  is known. Then the posterior mean  
 $\bar{\beta} = \bar{\mathbf{V}}[\{(X'X)^{-1}\}^{-1}\hat{\beta}_{\text{OLS}} + \underline{\mathbf{V}}^{-1}\underline{\beta}_{\text{prior}}]$ , where the posterior variance  $\bar{\mathbf{V}}$  is the inverse of the sum of the sample and prior precision matrices. Suppose we have a sample with  $(y, x)$  taking values such that in a model with intercept

$$X'X = \begin{bmatrix} 4 & 8 \\ 8 & 18 \end{bmatrix}; \quad (X'X)^{-1} = \frac{1}{4} \begin{bmatrix} 9 & -4 \\ -4 & 2 \end{bmatrix}; \quad X'y = \begin{bmatrix} 16 \\ 36 \end{bmatrix}$$

It will be easiest to do computations using matrix software in Mata or Stata. Find the MLE of  $\beta$  and its variance matrix. Find the posterior mean and variance of  $\beta$  if the prior for  $\beta$  is  $N(\mathbf{0}, 0.01 \times \mathbf{I})$ . Find the posterior mean and variance of  $\beta$  if the prior for  $\beta$  is  $N(\mathbf{0}, 10 \times \mathbf{I})$ . Explain what we learn about the role of the strength of the prior on  $\beta$ . Find the posterior mean and variance of  $\beta$  if the prior for  $\beta$  is  $N(\mathbf{0}, 0.01 \times \mathbf{I})$ , but we now have 100 times as many observations, so  $X'X$  and  $X'y$  are 100 times larger. Explain what we learn about the role of the sample size. Now, suppose the prior for  $\beta$  is instead uniform on  $\beta$ . Show that the posterior for  $\beta$  is the normal with mean  $(X'X)^{-1}X'y$  and variance  $(X'X)^{-1}$ . (Hint: Use the result that

$$(y - X\beta)'(y - X\beta) = \hat{\mathbf{u}}'\hat{\mathbf{u}} + (\beta - \hat{\beta})'X'X(\beta - \hat{\beta}), \text{ where } \hat{\beta} = (X'X)^{-1}X'y \text{ and } \hat{\mathbf{u}} = y - X\hat{\beta}.$$



# **Chapter 30**

## **Bayesian methods: Markov chain Monte Carlo algorithms**

## 30.1 Introduction

Bayesian methods were introduced in chapter [29](#), where focus was on the linear regression model and estimation used the `bayes` prefix and the `bayesmh` command.

In this chapter, we provide greater detail on Bayesian Markov chain Monte Carlo (MCMC) methods, with application to the probit model. We begin by using the `bayesmh` command with a user-defined likelihood. We then obtain MCMC draws from the posterior distribution for the probit model with flat prior using basic Mata code, rather than using the `bayesmh` command. Two MCMC methods are illustrated—the random-walk Metropolis–Hastings (MH) algorithm and the Gibbs sampler with data augmentation. The latter method is possible for models such as probit and tobit that are based on underlying latent variables.

The chapter concludes with a stand-alone treatment of multiple imputation of missing data. Multiple imputation is not necessarily Bayesian, and the initial discussion of missingness concepts and estimation and inference with multiply imputed datasets is relevant regardless of the imputation method used. A data example then presents one common method for imputation that uses MCMC draws based on data augmentation.

## 30.2 User-provided log likelihood

This section illustrates use of the `llevaluator()` option, which allows use of the `bayesmh` command when the log-likelihood function is specified by the user. The same generated data are used throughout the chapter.

### 30.2.1 Data and probit maximum-likelihood estimator

Let the data on  $N$  observations be denoted  $(\mathbf{y}, \mathbf{X})$ , where  $\mathbf{y}$  is data on the dependent variables and  $\mathbf{X}$  is data on exogenous regressors. And let  $\boldsymbol{\theta}$  denote the  $K$  parameters of the conditional density of  $\mathbf{y}$  given  $\mathbf{X}$ .

For the probit model, the dependent variable  $y_i$ , given  $K$  regressors  $\mathbf{x}_i$  that include an intercept, takes value 0 or 1 with

$$\begin{aligned}\Pr(y_i = 1 | \mathbf{x}_i, \boldsymbol{\beta}) &= \Phi(\mathbf{x}'_i \boldsymbol{\beta}) \\ \Pr(y_i = 0 | \mathbf{x}_i, \boldsymbol{\beta}) &= 1 - \Phi(\mathbf{x}'_i \boldsymbol{\beta})\end{aligned}\tag{30.1}$$

where  $\boldsymbol{\beta}$  is a  $K \times 1$  parameter vector and  $\Phi(\cdot)$  is the standard normal cumulative distribution function.

We generate data using the latent variable formulation for the probit model

$$\begin{aligned}y_i^* &= \mathbf{x}'_i \boldsymbol{\beta} + u_i \\ u_i &\sim N(0, 1) \\ y_i &= \begin{cases} 1 & \text{if } y_i^* > 0 \\ 0 & \text{if } y_i^* \leq 0 \end{cases}\end{aligned}\tag{30.2}$$

The following data-generating process (DGP) for a sample of size 100 sets the intercept to 0.5 and the slope to 1.0, and the single regressor is standard normally distributed.

```

. * Generate data N = 100 Pr[y=1|x] = PHI(0.5 + 1.0*x) and x ~ N(0,1)
. set obs 100
Number of observations (_N) was 0, now 100.
. set seed 1234567
. generate x = rnormal(0,1)
. generate ystar = 0.5 + 1*x + rnormal(0,1)
. generate y = (ystar > 0)
. generate cons = 1 // Mata code below requires a regressor for the intercept
. summarize

 Variable | Obs Mean Std. dev. Min Max
 | 100 -.1477064 1.003931 -2.583632 2.350792
 ystar | 100 .2901163 1.46373 -3.372719 3.316435
 y | 100 .59 .4943111 0 1
 cons | 100 1 0 1 1

. save mus230bayesgenerated, replace
(file mus230bayesgenerated.dta not found)
file mus230bayesgenerated.dta saved

```

The maximum likelihood estimator (MLE) is obtained using the `probit` command.

```

. * Fit probit model by MLE
. probit y x, nolog

Probit regression Number of obs = 100
 LR chi2(1) = 42.67
 Prob > chi2 = 0.0000
Log likelihood = -46.350193 Pseudo R2 = 0.3152

 y | Coefficient Std. err. z P>|z| [95% conf. interval]
 x | 1.137895 .2236915 5.09 0.000 .6994677 1.576322
 _cons | .4810185 .1591173 3.02 0.003 .1691543 .7928827

```

The ML estimates of the intercept and slope are, respectively, 0.481 and 1.138 compared with the DGP values of 0.5 and 1.0.

### 30.2.2 The `bayesmh` command

We specify a flat prior for the regression parameters, so all values of  $\beta$  are equally likely,

$$\pi(\beta) \propto 1$$

Even though this prior density is improper, the consequent posterior density is proper.

This model can be directly fit using command `bayesmh`, introduced in section [29.3.7](#). We obtain

```
. * Fit probit model by command bayesmh
. bayesmh y x, likelihood(probit) prior({y:_cons x}, flat) rseed(10101)
Burn-in ...
Simulation ...
Model summary

Likelihood:
y ~ probit(xb_y)
Prior:
{y:_cons x} ~ 1 (flat) (1)
```

(1) Parameters are elements of the linear form `xb_y`.

|                                          |                  |   |        |
|------------------------------------------|------------------|---|--------|
| Bayesian probit regression               | MCMC iterations  | = | 12,500 |
| Random-walk Metropolis-Hastings sampling | Burn-in          | = | 2,500  |
|                                          | MCMC sample size | = | 10,000 |
|                                          | Number of obs    | = | 100    |
|                                          | Acceptance rate  | = | .2081  |
|                                          | Efficiency: min  | = | .09261 |
|                                          | avg              | = | .104   |
| Log marginal-likelihood = -47.855029     | max              | = | .1154  |

| y     | Mean     | Std. dev. | MCSE    | Median   | Equal-tailed         |          |
|-------|----------|-----------|---------|----------|----------------------|----------|
|       |          |           |         |          | [95% cred. interval] |          |
| x     | 1.172479 | .2315767  | .006817 | 1.155511 | .7693384             | 1.644086 |
| _cons | .4912771 | .1649868  | .005421 | .4913284 | .1694699             | .8135934 |

The posterior means of the intercept and slope parameters are 0.491 and 1.172, close to the ML estimates of 0.481 and 1.138. The corresponding posterior standard deviations are 0.165 and 0.232, compared with the ML standard errors of 0.159 and 0.224. We expect some difference both because of randomness in the MH algorithm, and because even with a flat prior, the Bayesian results will differ from the ML results in finite samples.

### 30.2.3 The `bayesmh` command with user-provided evaluator

The `llevuator()` option of the `bayesmh` command enables the user to provide the likelihood function in cases where these are not already available as options.

The evaluator function is defined in a program that has as arguments `lnf`, the log-likelihood function, and any parameters of the model. Single-indexes such as  $\mathbf{x}'\boldsymbol{\beta}$  are treated as a single argument `xb`, similar to the programs for the `m1` command.

The log-likelihood function, given independence over  $i$ , is the sum of the log densities for each observation. For the probit model defined in (30.1), the log density for the  $i$ th observation is

$$\ln(y_i|\boldsymbol{\beta}, \mathbf{x}_i) = \begin{cases} \ln\{\Phi(\mathbf{x}'_i\boldsymbol{\beta})\} & \text{if } y_i^* > 0 \\ \ln\{1 - \Phi(\mathbf{x}'_i\boldsymbol{\beta})\} & \text{if } y_i^* \leq 0 \end{cases}$$

Program `probitll` defines the corresponding log-likelihood function.

```
. * Define evaluator function for probit model for input to command bayesmh
. program probitll
1. version 17
2. args lnf xb
3. tempvar lnfj
4. qui generate double `lnfj' = ln(normal(`xb')) if $MH_y == 1
5. qui replace `lnfj' = ln(normal(-`xb')) if $MH_y == 0
6. qui summarize `lnfj', meanonly
7. scalar `lnf' = r(sum)
8. end
```

The official option `likelihood(probit)` for the `bayesmh` command is then replaced by option `llevuator(probitll)`, where program `probitll` defined the log-likelihood function. The prior, here a flat prior for  $\boldsymbol{\beta}$ , is defined in the usual way for the `bayesmh` command. This yields

```

. * Fit probit model by command bayesmh with user-provided evaluator
. bayesmh y x, llevaluator(probitll) prior({y:_cons x}, flat) rseed(10101)
Burn-in ...
Simulation ...
Model summary

```

---

Likelihood:  
 $y \sim \text{probitll}(xb\_y)$

Prior:  
 $\{y:_\text{cons} x\} \sim 1 \text{ (flat)}$

---

(1)

(1) Parameters are elements of the linear form  $xb\_y$ .

|                                          |                  |   |        |
|------------------------------------------|------------------|---|--------|
| Bayesian regression                      | MCMC iterations  | = | 12,500 |
| Random-walk Metropolis-Hastings sampling | Burn-in          | = | 2,500  |
|                                          | MCMC sample size | = | 10,000 |
|                                          | Number of obs    | = | 100    |
|                                          | Acceptance rate  | = | .1993  |
|                                          | Efficiency: min  | = | .1045  |
|                                          | avg              | = | .1145  |
|                                          | max              | = | .1245  |

Log marginal-likelihood = -47.877531

| y     | Mean     | Std. dev. | MCSE    | Median   | Equal-tailed         |          |
|-------|----------|-----------|---------|----------|----------------------|----------|
|       |          |           |         |          | [95% cred. interval] |          |
| x     | 1.174587 | .2294219  | .006502 | 1.170978 | .7509662             | 1.654543 |
| _cons | .4919817 | .1611742  | .004985 | .4946297 | .1892769             | .8195851 |

The efficiency and the acceptance rate are very similar to those for the example in section [30.2.2](#), which used option `likelihood(probit)`. The posterior means of the intercept and slope parameters are 0.492 and 1.175, compared with 0.491 and 1.172. The corresponding posterior standard deviations are 0.161 and 0.229, compared with 0.165 and 0.232.

These small differences arise because option `llevaluator()` by default sets the initial values of the parameters to zero, whereas option `likelihood(probit)` uses different starting values. Alternative starting values can be specified using, for example, the option `initial({y:_cons} 0.5 {y:x} 1.2)`.

The manual entry [BAYES] **bayesmh evaluators** provides many examples. These include models with additional parameters, as well as additional coding in the program defining the evaluator to handle missing observations, using temporary marker variable `$MH_touse`.

The `evaluator()` option of the `bayesmh` command enables the user to provide both the likelihood function and the prior density for the parameters by defining the sum of the log likelihood and the prior density. Because command `bayesmh` provides a wide range of priors, including flat priors, there is generally less need to use this option; an exception would be providing the Jeffreys prior because this prior is determined by the likelihood function.

## 30.3 MH algorithm in Mata

In this section, we repeat the preceding example but directly code the random-walk MH algorithm in Mata rather than use the `bayesmh` command.

### 30.3.1 The log posterior

The likelihood function for the probit model can be written as

$$L(\mathbf{y}|\boldsymbol{\beta}, \mathbf{X}) = \prod_{i=1}^N \Phi(\mathbf{x}'_i \boldsymbol{\beta})^{y_i} \{1 - \Phi(\mathbf{x}'_i \boldsymbol{\beta})\}^{1-y_i}$$

where  $\Phi(\cdot)$  is the standard normal distribution function. We use a flat uninformative prior, so

$$\pi(\boldsymbol{\beta}) \propto 1$$

Combining the likelihood and the prior, we see the posterior is

$$\begin{aligned} p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}) &\propto L(\mathbf{y}|\boldsymbol{\beta}, \mathbf{X}) \times \pi(\boldsymbol{\beta}) \\ &\propto \prod_{i=1}^N \Phi(\mathbf{x}'_i \boldsymbol{\beta})^{y_i} \{1 - \Phi(\mathbf{x}'_i \boldsymbol{\beta})\}^{1-y_i} \times 1 \\ &\propto \prod_{i=1}^N \Phi(\mathbf{x}'_i \boldsymbol{\beta})^{y_i} \{1 - \Phi(\mathbf{x}'_i \boldsymbol{\beta})\}^{1-y_i} \end{aligned}$$

Note that the posterior  $p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{X})$  is defined only up to a scale factor.

The log posterior is then

$$\ln p(\boldsymbol{\beta} | \mathbf{y}, \mathbf{X}) \propto \sum_{i=1}^N [y_i \times \ln \{\Phi(\mathbf{x}'_i \boldsymbol{\beta})\} + (1 - y_i) \times \ln \{1 - \Phi(\mathbf{x}'_i \boldsymbol{\beta})\}]$$

### 30.3.2 Random-walk Metropolis algorithm

The random-walk MH algorithm is presented in section [29.3.3](#). Here we apply the algorithm to the probit model and implement the algorithm in Mata.

At the  $s$ th round, the proposal distribution is the  $N(\boldsymbol{\beta}^{(s-1)}, c^2 \mathbf{I})$  distribution for specified  $c$ . So  $\boldsymbol{\beta}_s^* = \boldsymbol{\beta}^{(s-1)} + \mathbf{v}$ , where  $\mathbf{v}$  is a draw from the  $N(\mathbf{0}, c^2 \mathbf{I})$  distribution.

The acceptance rule is determined based in part by the ratio of posteriors,

$$r_s = p(\boldsymbol{\beta}_s^*) / p(\boldsymbol{\beta}^{(s-1)})$$

We then set  $\boldsymbol{\beta}_s = \boldsymbol{\beta}^*$  if  $u_s < r_s$  or  $\boldsymbol{\beta}_s = \boldsymbol{\beta}_{s-1}$  if  $u_s > r_s$ , where  $u_s$  is a draw from the uniform(0, 1) distribution. Taking logs, the acceptance rule is equivalent to setting  $\boldsymbol{\beta}_s = \boldsymbol{\beta}_s^*$  if  $\ln u_s < \ln r_s$ .

For the probit posterior with flat prior defined above, we have

$$\begin{aligned} \ln r_s &= \sum_i [y_i \ln \Phi(\mathbf{x}'_i \boldsymbol{\beta}^*) + (1 - y_i) \ln \{1 - \Phi(\mathbf{x}'_i \boldsymbol{\beta}^*)\}] \\ &\quad - \sum_i \left[ y_i \ln \Phi(\mathbf{x}'_i \boldsymbol{\beta}^{(s-1)}) + (1 - y_i) \ln \left\{ 1 - \Phi(\mathbf{x}'_i \boldsymbol{\beta}^{(s-1)}) \right\} \right] \end{aligned}$$

Then  $\boldsymbol{\beta}_s = \boldsymbol{\beta}_s^*$  if  $\ln u_s < \ln r_s$ ; otherwise,  $\boldsymbol{\beta}_s = \boldsymbol{\beta}_{s-1}$ .

### 30.3.3 Numerical example in Mata

We first define some global macros. The first 10,000 MH draws will be discarded (burn-in), and the next 10,000 draws kept. For draws from

$N(\mathbf{0}, c^2 \mathbf{I})$ , we set  $c = 0.25$ .

```
. * Define globals for number of reps and the key tuning parameter
. global s1 10000 // Number of retained reps
. global s0 10000 // Number of burnin reps
. global sdscale 0.25 // May need to change c in proposal b + $sdscale*N(0,I)
```

The key tuning parameter is the constant  $c$ , which in the current example is set to  $c = 0.25$ . A rationale for this value is that ideally the proposal distribution is close to the posterior distribution. With a flat prior, the posterior distribution will be close to the log likelihood, and the ML estimates of the two model parameters had standard deviations of 0.22 and 0.16.

The `bayesmh` command initially sets  $c = 2.38/K$ , following a suggestion made in several articles, where  $K$  is the number of model parameters in an MH block. In the current example, with  $c = 2.38/2 = 1.19$ , the chain diverged. The `bayesmh` command includes a rule for changing the value of  $c$  every 100 draws (the default), so it is more likely to overcome a poor initial value of  $c$ . And option `scale()` allows specification of a different initial value for  $c$ .

The following lengthy Mata code, based on the MATLAB code of [Koop \(2003\)](#), implements the MH algorithm.

```

. * Mata to obtain the posterior draws of b for probit MH algorithm
. set seed 10101
. mata
: mata (type end to exit)
:
: // (1) Create y vector and X matrix from Stata dataset using st_view()
: st_view(y=., ., "y") // Dependent
: st_view(X=., ., ("cons", "x")) // Regressors
: Xnames = ("cons", "x") // Used to label output
: // Calculate a few quantities outside the loop for later use
: n = rows(X)
: k = cols(X)
: ones = J(n,1,1)
: // Specify the number of replications
: s0 = $s0 // Number of burnin reps
: s1 = $s1 // Number of retained reps
: s = s0+s1 // Total reps
: // Store all draws and MH acceptance rate in the following matrices
: b_all = J(k,s1,0)
: accept_all = J(1,s1,0)
: // Initialization
: bdraw = J(k,1,0) // Starting b value is vector of zeros
: lpostdraw = -1*10^10 // Starting value of ln(posterior) is small
: // So accept initial MH draw
:
: // (2) Now do MH loop and make the posterior draws
: // Begin MH loop
: for (irep=1; irep<=s; irep++) {
: // Draw new candidate value of b from MH random-walk chain
: bcandidate = bdraw + $sdscale*rnormal(k,1,0,1)
: // Note: For different data, you may need to change the global sdscale
: // Best is bcandidate = bdraw + z, where z ~ N(0, post. variance of b)
: // Compute the log-posterior at the candidate value of b
: // The assumed prior for b is uninformative
: // so the posterior is proportional to the usual probit likelihood
: probitprob = normal(X*bcandidate)
: lpostcandidate = ones`*(y:*ln(probitprob) +(ones-y):*ln(ones-probitprob))
: // Accept the candidate draw on basis of posterior probability ratio
: // if uniform > (posterior(bcandidate) / posterior(bdraw))
: // where bcandidate is current b and bdraw is previous b
: // Taking logs the rule is the same as
: // if ln(uniform) > (lpostcandidate - lpostdraw)
: laccprobability = lpostcandidate - lpostdraw
: accept = 0
: if (ln(uniform(1,1)) < laccprobability) {
: lpostdraw = lpostcandidate
: bdraw = bcandidate
: accept = 1
: }
:
```

```

> // Store the draws after burn-in of b and whether accept draw
> if (irep>s0) {
> // After discarding burn-in, store all draws
> j = irep-s0
> b_all[.,j] = bdraw // These are the posterior draws
> accept_all[.,j] = accept // These are one if new draw accepted
> }
> }
:
: // End MH loop
:
:
: // (3) Pass results back to Stata
: // The next command is needed for conformability. It requires $s1 > N
: stata("set obs $s1")
Number of observations (_N) was 100, now 10,000.
:
: accept = accept_all'
:
: st_addvar("byte", "accept")
5
:
: st_store(., "accept", accept)
:
: beta = b_all'
:
: // Loop sends each column of beta to Stata as separate variable beta_i
: for (i=1; i<=k; i++) {
> v = beta[.,i]
> vname = "beta" + strofreal(i)
> st_addvar("double", vname)
> st_store(., vname, v)
> }
6
7
:
: end

```

---

The Mata code involves three segments. The first segment, denoted (1), reads in the data and initializes key constants and matrices.

The second segment, denoted (2), implements the MH algorithm. Matrix `bdraw` contains the previous draw  $\beta_{s-1}$ , matrix `bcandidate` contains the proposal draw  $\beta_s^*$ , and if the proposal draw is accepted, then matrix `bdraw` is updated.

The third segment, denoted (3), passes the matrices `beta` and `accept` back to Stata as variables because some of the subsequent analysis is easier to do in Stata than in Mata. The code uses Mata functions `st_addvar()` and `st_store()`. Mata function `strofreal()` is used to convert the real number index `i` to a string variable "`i`" so that, for example, a variable named `beta2` is created when `i=2`.

It would be much simpler to directly pass the matrices to Stata using Mata command `st_matrix("beta", beta)`, and then in Stata to give command `svmat(beta)` to convert a matrix to variables. The reason for not taking this simpler approach is that while matrices in Mata can be very large, in Stata their size is restricted. For Stata/BE, the limit is  $800 \times 800$ , and even Stata/SE and Stata/MP restrict matrix size in Stata 17 to be no more than, respectively,  $11000 \times 11000$  and  $65534 \times 65534$ .

Command `summarize` provides a summary of the posterior draws and acceptance rate, and command `centile` gives the 95% Bayesian credible region.

| Variable                            | Obs    | Mean       | Std. dev. | Min                                    | Max      |
|-------------------------------------|--------|------------|-----------|----------------------------------------|----------|
| beta1                               | 10,000 | .4868512   | .1573034  | -.0620936                              | 1.173329 |
| beta2                               | 10,000 | 1.167617   | .2263273  | .4205724                               | 2.051684 |
| accept                              | 10,000 | .4302      | .4951287  | 0                                      | 1        |
| . centile beta2, centile(2.5, 97.5) |        |            |           |                                        |          |
| Variable                            | Obs    | Percentile | Centile   | Binom. interp.<br>[95% conf. interval] |          |
| beta2                               | 10,000 | 2.5        | .7456415  | .726276                                | .7578553 |
|                                     |        | 97.5       | 1.631214  |                                        |          |

Compared with the results from the `bayesmh` command given in section [30.2.2](#), the posterior means of the intercept and slope parameters are 0.487 and 1.168, rather than 0.491 and 1.173. The corresponding posterior standard deviations are 0.157 and 0.226, compared with 0.165 and 0.232. The 95% Bayesian credible region for the slope parameter is [0.746, 1.631] compared with [0.769, 1.644].

Because the MH posterior draws are correlated, the 10,000 retained draws convey less precision than 10,000 independent draws. This loss is measured by the efficiency, the reciprocal of the sum of the squared autocorrelation coefficients of the draws;  $\text{eff} = 1 / \{1 + 2 \times \sum_{j=1}^J (1 + \hat{\rho}_j^2)\}$ .

To compute the autocorrelations to  $J = 100$  and hence the efficiency, we use the time-series commands in Stata as follows.

```

. * Compute the efficiency of the MH algorithm
. generate s = _n
. tsset s
Time variable: s, 1 to 10000
Delta: 1 unit
. qui ac beta2, lags(100) gen(ac)
. qui generate ac_sq = ac^2
. qui summarize ac_sq
. di "Efficiency = " 1/(1+2*r(sum))
Efficiency = .18277753

```

The efficiency of 0.183 for  $\beta_2$  is good.

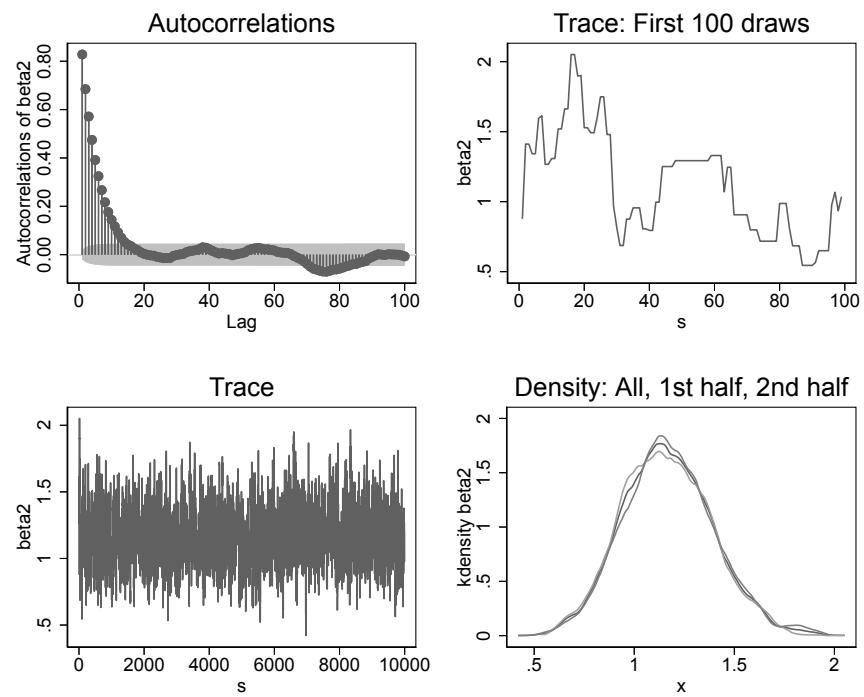
We next construct various diagnostics graphs for  $\beta_2$ , similar to those produced by the `bayesgraph diagnostics` command.

```

. * Plot various diagnostics for the posterior draws of b2
. qui ac beta2, title("Autocorrelations") lags(100)
> note(" ", ring(0) pos(3))
. qui line beta2 s if s < 100, title("Trace: First 100 draws")
. qui line beta2 s, title("Trace")
. qui graph twoway (kdensity beta2) (kdensity beta2 if s<=5000)
> (kdensity beta2 if s>5000), title("Density: All, 1st half, 2nd half")
> legend(off) note(" ", ring(0) pos(3))

```

From figure 30.1, the diagnostic graphs all look good. The autocorrelations die out fast, many of the first 100 draws after the burn-in draws are accepted, the trace for all 10,000 retained draws shows no pattern, and the densities for the first and second halves of the retained draws are similar.



**Figure 30.1.** Diagnostics for posterior draws of education coefficient

## 30.4 Data augmentation and the Gibbs sampler in Mata

For Bayesian MCMC methods, it is usually advantageous to use the Gibbs sampler, presented in sections [29.3.5](#) and [29.7.2](#), wherever possible. However, to do so requires analytical results for some of or all the conditional posteriors.

Analytical results are most readily available for the linear regression model under normality; some of these results were given in section [29.5](#). Several leading nonlinear regression models, notably, tobit, binary probit, and multinomial probit, are models based on latent variables that follow a linear normal model. As explained next, data augmentation methods enable application of the Gibbs sampler to these latent variable models.

### 30.4.1 Data augmentation

Latent variable models, such as probit and tobit models, can be modeled as observing  $y_1, \dots, y_N$  based on latent variables  $y_1^*, \dots, y_N^*$ . Furthermore, for many of these models,  $y_1^*, \dots, y_N^*$  completely determines  $y_1, \dots, y_N$ . For example, for the probit model,  $y_i = \mathbf{1}(y_i^* > 0)$ .

For simplicity, we initially suppress conditioning on regressors  $\mathbf{X}$  and consider the general case with parameters  $\boldsymbol{\theta}$ . The desired posterior is  $p(\boldsymbol{\theta}|\mathbf{y})$ . Data augmentation expands this posterior to include the latent variables  $\mathbf{y}^* = (y_1^*, \dots, y_N^*)'$ . The goal then is to make draws from the augmented posterior  $p(\boldsymbol{\theta}, \mathbf{y}^*|\mathbf{y})$ . Given draws of  $\boldsymbol{\theta}$  and  $\mathbf{y}^*$  from the joint posterior, the draws of  $\mathbf{y}^*$  are ignored, while the draws of  $\boldsymbol{\theta}$  are draws from the desired posterior  $p(\boldsymbol{\theta}|\mathbf{y})$  because  $p(\boldsymbol{\theta}|\mathbf{y})$  is the marginal posterior of  $\boldsymbol{\theta}$  with respect to the joint posterior of  $p(\boldsymbol{\theta}, \mathbf{y}^*|\mathbf{y})$ .

The draws from the augmented posterior  $p(\boldsymbol{\theta}, \mathbf{y}^*|\mathbf{y})$  can be obtained using the Gibbs sampler, with alternating draws from the conditional posterior  $p(\boldsymbol{\theta}|\mathbf{y}^*, \mathbf{y})$  and the conditional posterior  $p(\mathbf{y}^*|\boldsymbol{\theta}, \mathbf{y})$ , assuming that these conditional posteriors are known.

### 30.4.2 Gibbs sampler data-augmentation algorithm for probit

Here we focus on the probit model. From (30.2), the latent variable is  $y_i^* = \mathbf{x}'_i \boldsymbol{\beta} + u_i$ , and we observe  $y_i = 1$  or 0 according to whether  $y_i^* \geq 0$ .

If the latent variable  $y_i^*$  was observed and a flat prior is used for  $\boldsymbol{\beta}$ , then Bayesian analysis is straightforward. Because  $y_i^* \sim N(0, 1)$ , the log likelihood is  $\ln p(\boldsymbol{\beta} | \mathbf{y}^*, \mathbf{X}) = -(N/2) \ln(2\pi) - (1/2) \sum_{i=1}^N (y_i^* - \mathbf{x}'_i \boldsymbol{\beta})^2$ . Given prior density  $\pi(\boldsymbol{\beta}) \propto 1$ , the log-posterior density is actually known and is

$$\ln p(\boldsymbol{\beta} | \mathbf{y}^*, \mathbf{X}) = -\frac{N}{2} \ln(2\pi) - \frac{1}{2} \sum_{i=1}^N (y_i^* - \mathbf{x}'_i \boldsymbol{\beta})^2$$

Of course, the latent variable is not observed. The data-augmentation method treats the latent variables as  $N$  additional parameters  $y_1^*, \dots, y_N^*$  and makes random draws of  $y_1^*, \dots, y_N^*$ .

Specifically, given knowledge of  $\boldsymbol{\beta}$  and the data, the distribution of these additional parameters is known because if  $y_i = 1$ , then  $y_i^* = \mathbf{x}'_i \boldsymbol{\beta} + u_i \geq 0$ . So  $y_i^*$  is a draw from the  $N(\mathbf{x}'_i \boldsymbol{\beta}, 1)$  distribution truncated from below at 0. Similarly, if  $y_i = 0$ , then  $y_i^*$  is a draw from the  $N(\mathbf{x}'_i \boldsymbol{\beta}, 1)$  distribution truncated from above at 0.

The conditional posterior for the parameters  $\boldsymbol{\beta}$  given the latent variables,  $p(\boldsymbol{\beta} | \mathbf{y}, \mathbf{y}^*, \mathbf{X}) = p(\boldsymbol{\beta} | \mathbf{y}^*, \mathbf{X})$ ; conditioning on  $\mathbf{y}$  in addition to  $\mathbf{y}^*$  is irrelevant because knowledge of  $\mathbf{y}^*$  subsumes knowledge of  $\mathbf{y}$ . The conditional posterior for the parameters  $\boldsymbol{\beta}$  given the latent variables is  $p(\boldsymbol{\beta} | \mathbf{y}, \mathbf{y}^*, \mathbf{X})$ . This simplifies to  $p(\boldsymbol{\beta} | \mathbf{y}^*, \mathbf{X})$  because conditioning on  $\mathbf{y}$  in addition to  $\mathbf{y}^*$  is irrelevant because knowledge of  $\mathbf{y}^*$  subsumes knowledge of  $\mathbf{y}$ .

The  $s$ th round of the Gibbs sampler draws  $\boldsymbol{\beta}_s$  from  $p(\boldsymbol{\beta}_s | y_{1,s-1}^*, \dots, y_{N,s-1}^*, \mathbf{y}, \mathbf{X})$  and then draws  $y_{1,s}^*, \dots, y_{N,s}^*$  from  $p(y_1^*, \dots, y_N^* | \boldsymbol{\beta}_s, \mathbf{y}, \mathbf{X})$ .

### 30.4.3 Numerical example

We continue with the same probit example as estimated previously using the random-walk MH algorithm.

We first define some global macros.

```
. * Define globals for number of reps and the key tuning parameter
. use mus230bayesgenerated, clear
. global s1 10000 // Number of retained reps
. global s0 10000 // Number of burnin reps
```

The following Mata code, based on the MATLAB code of [Koop \(2003\)](#), implements the Gibbs sampler with data augmentation.

```

. set seed 10101
. mata
: mata (type end to exit)
:
: // (1) Create y vector and X matrix from Stata dataset using st_view
: st_view(y=., ., "y") // Dependent
: st_view(X=., ., ("cons", "x")) // Regressors
: Xnames = ("cons", "x") // Used to label output
: // Calculate a few quantities outside the loop for later use
: n = rows(X)
: k = cols(X)
: Xsquare = cross(X,X)
: Xtxinv = invsym(Xsquare)
: Xtxinvchol = cholesky(Xtxinv)
: v1 = n
: // Specify the number of replications
: s0 = $s0 // Number of burnin reps
: s1 = $s1 // Number of retained reps
: s = s0+s1 // Total reps
: // Store all draws in the following matrices
: y1 = J(s0+s1,1,0)
: b_all = J(k,s1,0)
: // Prior for beat is noninformative
: // Choose a starting value for latent data
: ystar = y
:
: // (2) Now, do Gibbs sampler loop and make the posterior draws
: for (irep=1; irep<=s; irep++) {
: // Posterior step: draw from beta | y* ~ N[bols*, (X`X)^-1]
: // This is using noninformative prior for beta
: bols = Xtxinv*cross(X,ystar)
: b1 = bols
: bdraw = b1 + Xtxinvchol*rnormal(k,1,0,1) // invnormal(uniform(k,1))
: // Imputation step: make one draw of vector ystar
: // where for ith observation ystar_i | y,b is truncated normal
: // Right: If y = 1, we need draw from truncated N[0,1] with ystar > -mu
: // Left: If y = 0, we need draw from truncated N[0,1] with ystar < -mu
: for (i=1; i<=n; i++) {
: mu = X[i,.]*bdraw
: if (y[i,1]==0) {
: uright = normal(-mu)*uniform(1,1)
: ystar[i,1] = mu + invnormal(uright)
: }
: else {
: uleft = normal(-mu) + (1-normal(-mu))*uniform(1,1)
: ystar[i,1] = mu + invnormal(uleft)
: }
: }
: // Store the draws of b after burn-in plus a diagnostic used in Koop
: if (irep>s0) {
: // After discarding burn-in, store all draws
: j = irep-s0
: b_all[.,j] = bdraw // These are the posterior draws
: }
: }
: // End Gibbs loop
:
: // (3) Pass results back to Stata
: // The next command is needed for conformability. It requires $s1 > N
: stata("set obs $s1")
Number of observations (_N) was 100, now 10,000.
: beta = b_all'
:
: // Loop sends each column of beta to Stata as separate variable beta_i
: for (i=1; i<=k; i++) {
: v = beta[.,i]
: vname = "beta" + strofreal(i)
: st_addvar("double", vname)
: st_store(., vname, v)
: }
5
6
: end

```

---

The first and third segments are essentially the same as for the MH algorithm example in section [30.3.3](#). The second segment, labeled (2), implements the Gibbs sampler.

We obtain the following results.

```
. * Analyze the posterior draws from probit Gibbs sampler algorithm
. summarize beta*
```

| Variable | Obs    | Mean     | Std. dev. | Min       | Max      |
|----------|--------|----------|-----------|-----------|----------|
| beta1    | 10,000 | .4842779 | .1582509  | -.2082249 | 1.158187 |
| beta2    | 10,000 | 1.173542 | .2259341  | .4758451  | 2.346783 |

| Variable | Obs    | Percentile | Centile  | Binom. interp.       |          |
|----------|--------|------------|----------|----------------------|----------|
|          |        |            |          | [95% conf. interval] |          |
| beta2    | 10,000 | 2.5        | .7512463 | .7392763             | .7636364 |
|          |        | 97.5       | 1.646951 | 1.629367             | 1.660316 |

Compared with the results from the `bayesmh` command given in section [30.2.2](#), the posterior means of the intercept and slope parameters are 0.484 and 1.174, rather than 0.491 and 1.172. The corresponding posterior standard deviations are 0.158 and 0.226, compared with 0.165 and 0.232. The acceptance rate is automatically one for this method that uses only the Gibbs sampler throughout. The 95% Bayesian credible region for the slope parameter is [0.751, 1.647] compared with [0.769, 1.644].

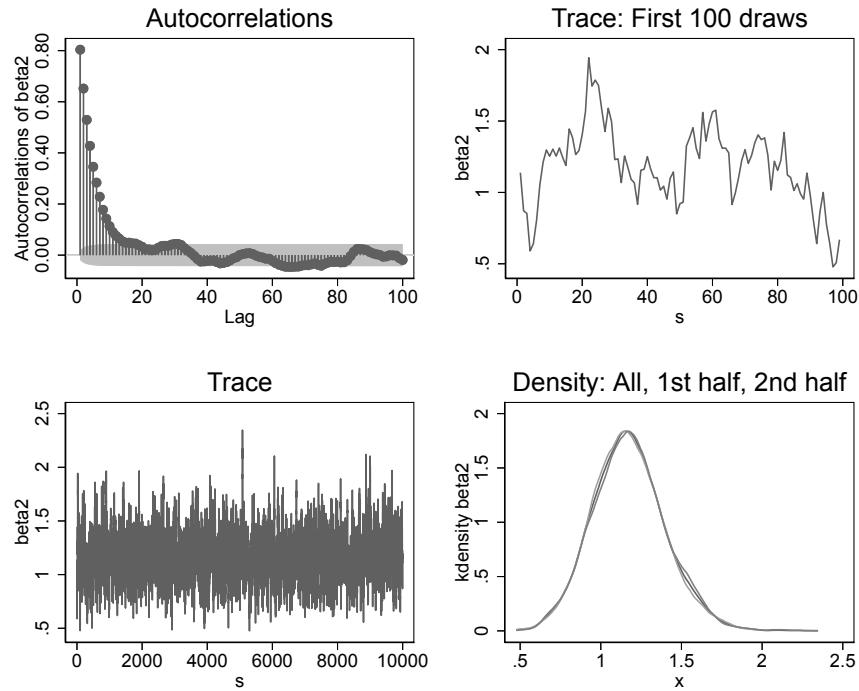
We again compute the efficiency of the method.

```
. * Compute the efficiency of the Gibbs sampler algorithm
. generate s= _n
. tsset s
Time variable: s, 1 to 10000
 Delta: 1 unit
. qui ac beta2, lags(100) gen(ac)
. qui generate ac_sq = ac^2
. qui summarize ac_sq
. di "Efficiency = " 1/(1+2*r(sum))
Efficiency = .20486695
```

The efficiency of 0.205 is quite similar to the efficiency obtained using the MH algorithm.

We again construct various diagnostics graphs for  $\beta_2$  similar to those produced by the `bayesgraph diagnostics` command.

```
. * Plot various diagnostics for the posterior draws of b2
. qui ac beta2, title("Autocorrelations") lags(100)
> note(" ", ring(0) pos(3))
. qui line beta2 s if s < 100, title("Trace: First 100 draws")
. qui line beta2 s, title("Trace")
. qui graph twoway (kdensity beta2) (kdensity beta2 if s<=5000)
> (kdensity beta2 if s>5000), title("Density: All, 1st half, 2nd half")
> legend(off) note(" ", ring(0) pos(3))
```



**Figure 30.2.** Diagnostics for posterior draws of education coefficient

The diagnostic plots all look good. The trace for the first 100 retained draws illustrates that a different posterior draw of the parameters is obtained at each round, so the acceptance rate is one.

### 30.4.4 Further examples

Data augmentation can be applied to a range of models, especially those based on latent normal variable models. A good reference is [Chib \(2001\)](#).

The preceding example with a flat prior can be modified to allow a normal prior for  $\beta$ , in which case the conditional posterior  $p(\beta|y^*, \mathbf{X})$  is also normal.

A more advanced example is the multinomial probit model. The random utility for the  $j$ th of  $m$  alternatives is  $U_{ij}^* = \mathbf{x}'_{ij}\beta + \varepsilon_{ij}$ , where the  $m \times 1$  error vector  $\varepsilon_i \sim N(\mathbf{0}, \Sigma_\varepsilon)$ . The  $j$ th alternative is chosen, so  $y_{ij} = 1$ , if  $U_{ij}^* > U_{ik}^*$  all  $k \neq j$ . A Bayesian analysis might specify a normal prior for  $\beta$  and a Wishart prior for  $\Sigma_\varepsilon^{-1}$ . Data augmentation introduces the latent utilities  $\mathbf{U}_i = (U_{i1}, \dots, U_{im})$  as additional parameters. Let

$\mathbf{U} = (\mathbf{U}_1, \dots, \mathbf{U}_N)$  and  $\mathbf{y} = (y_1, \dots, y_N)$ . The Gibbs sampler for the joint posterior  $p(\beta, \mathbf{U}, \Sigma_\varepsilon | \mathbf{y}, \mathbf{X})$  then cycles between (1) the conditional posterior for  $\beta | \mathbf{U}, \Sigma_\varepsilon, \mathbf{y}, \mathbf{X}$ ; (2) the conditional posterior for  $\Sigma_\varepsilon | \beta, \mathbf{U}, \mathbf{y}, \mathbf{X}$ ; and (3) the conditional posterior for  $\mathbf{U} | \beta, \Sigma_\varepsilon, \mathbf{y}, \mathbf{X}$ . See [McCulloch and Rossi \(1994\)](#) for details and application.

## 30.5 Multiple imputation

Imputation methods were briefly mentioned in sections 2.4.6 and [19.10.5](#). Here we consider multiple imputation, where missing observations are imputed several times.

### 30.5.1 Missingness mechanisms

Let  $\mathbf{W}$  denote an  $N \times p$  data matrix, let  $\mathbf{W}_{\text{obs}}$  denote the observed part of  $\mathbf{W}$ , and let  $\mathbf{W}_{\text{miss}}$  denote the missing part of  $\mathbf{W}$ . The goal of multiple imputation is to impute values of  $\mathbf{W}_{\text{miss}}$  from  $\mathbf{W}_{\text{obs}}$ . This requires specification of a stochastic model for  $\mathbf{W}$  or for components of  $\mathbf{W}$ . The strength of assumptions made varies with the way in which data are missing.

Data are missing at random (MAR) if the cause of missing data is unrelated to missing data, though is related to observed data. Let  $\mathbf{S}$  denote an  $N \times p$  matrix of indicators with elements of zero or one depending on whether corresponding values in  $\mathbf{W}$  are missing. Then data are MAR if  $\Pr(\mathbf{S}|\mathbf{W}_{\text{obs}}, \mathbf{W}_{\text{mis}}) = \Pr(\mathbf{S}|\mathbf{W}_{\text{obs}})$ . This situation arises in a regression context where analysis is conditional on regressors and the only missing data are those on exogenous regressors.

If instead the cause of missing data is unrelated to either observed data or missing data, then data are said to be missing completely at random (MCAR). Formally,  $\Pr(\mathbf{S}|\mathbf{W}_{\text{obs}}, \mathbf{W}_{\text{mis}}) = \Pr(\mathbf{S})$ . While consistent estimation and valid inference is then possible using only nonmissing data, imputing MCAR data can improve efficiency.

A more challenging situation is where the cause of the missing data is related to missing data. Then  $\Pr(\mathbf{S}|\mathbf{W}_{\text{obs}}, \mathbf{W}_{\text{mis}}) \neq \Pr(\mathbf{S}|\mathbf{W}_{\text{obs}})$ , and the data are said to be missing not at random (MNAR). This assumption applies in a regression context where data are missing on endogenous regressors or on the dependent variable.

Simple imputation methods such as replacing missing values with nonmissing mean values can lead to inconsistent estimation and invalid statistical inference; the chapter exercise 5 provides an example. Multiple imputation can avoid these problems, provided the model used to impute missing data is a good model. Multiple imputation is best used in situations where data are MAR or MCAR. In principle, one can also impute when data are MNAR, but then the necessary stochastic assumptions are much stronger.

### 30.5.2 Estimation and inference for multiple imputation

A single imputation creates data  $\mathbf{W}_{\text{misximp}}$  that are imputed values of the missing data  $\mathbf{W}_{\text{mis}}$ . Then the completed imputed dataset is

$\mathbf{W}_{\text{imp}} = (\mathbf{W}_{\text{obs}}, \mathbf{W}_{\text{misximp}})$ . Given  $\mathbf{W}_{\text{imp}}$ , we can obtain an estimate  $\hat{\boldsymbol{\theta}}$  of  $\boldsymbol{\theta}$  in the usual way. But regular statistical inference based on  $\hat{\boldsymbol{\theta}}$  will be invalid because it fails to account for the additional randomness caused by imputing  $\mathbf{W}_{\text{misximp}}$ .

Multiple imputation imputes the missing data  $m$  times, leading to  $m$  independent imputations  $\mathbf{W}_{\text{misximp},r}$ ,  $r = 1, \dots, m$  and hence  $m$  completed imputed datasets  $\mathbf{W}_{\text{imp},r}$ ,  $r = 1, \dots, m$ .

Each dataset is used to fit a model with parameter  $\boldsymbol{\theta}$ , leading to  $m$  estimates  $\hat{\boldsymbol{\theta}}_1, \dots, \hat{\boldsymbol{\theta}}_m$ . Each estimate  $\hat{\boldsymbol{\theta}}_r$  has two sources of randomness—the usual sampling variability measured by the usual estimated variances  $\widehat{\mathbf{V}}_r$  plus additional randomness due to imputation.

The parameter  $\boldsymbol{\theta}$  is estimated by  $\bar{\boldsymbol{\theta}} = 1/m \sum_{r=1}^m \hat{\boldsymbol{\theta}}_r$ , the average of the  $m$  estimates  $\hat{\boldsymbol{\theta}}_1, \dots, \hat{\boldsymbol{\theta}}_m$ . The variance of  $\bar{\boldsymbol{\theta}}$  is estimated by the sum of the average of  $\widehat{\mathbf{V}}_1, \dots, \widehat{\mathbf{V}}_m$  plus the variance in  $\hat{\boldsymbol{\theta}}_1, \dots, \hat{\boldsymbol{\theta}}_m$  across the  $m$  imputed samples. The latter quantity accounts for the additional variation due to imputation. Then the estimate  $\bar{\boldsymbol{\theta}}$  has estimated variance

$$\widehat{\mathbf{V}}(\bar{\boldsymbol{\theta}}) = \left( \frac{1}{m} \sum_{r=1}^m \widehat{\mathbf{V}}_r \right) + \frac{1 + (1/m)}{m - 1} \sum_{r=1}^m (\hat{\boldsymbol{\theta}}_r - \bar{\boldsymbol{\theta}})^2$$

where use of  $\{1 + (1/m)\}/(m - 1)$  rather than the more obvious  $1/(m - 1)$  is a finite-imputation correction.

The need to impute leads to an efficiency loss compared with having all data available. This efficiency loss is greater the smaller is  $m$  and the more data that are missing. Additionally, this efficiency loss varies with the model used and the particular data at hand. So there is no clear best value for  $m$ . A conservative choice for researchers doing their own imputations might be  $m = 20$ .

Many major individual-level datasets impute missing data for at least key variables such as income. Typically, there is only a single imputation. Some surveys instead publish several imputed datasets to enable users to control for the additional uncertainty due to imputation. Then low values such as  $m = 5$  are typically used.

### 30.5.3 Regression-based imputation

One method for data imputation is regression-based imputation for observations that are missing on a single variable.

We consider a simple example where data are missing for some observations of the variable  $x$  and are completely available for all observations of variables  $\mathbf{z}$ . Thus, a completed dataset would be  $\mathbf{W} = (\mathbf{x}, \mathbf{Z})$ . Partition missing and nonmissing observations on variable  $x$  as  $\mathbf{x} = (\mathbf{x}_o, \mathbf{x}_m)$  and correspondingly partition observations on variable  $\mathbf{z}$  as  $\mathbf{Z} = (\mathbf{Z}_o, \mathbf{Z}_m)$ .

Suppose  $x$  is a count that we believe can be well modeled parametrically as being Poisson distributed with mean  $\exp(\mathbf{z}'\boldsymbol{\beta})$ . Then regression-based imputation of  $\mathbf{x}_m$  is obtained as follows.

First, use nonmissing observations to perform Poisson regression of  $\mathbf{x}_o$  on  $\mathbf{Z}_o$ , giving estimate  $\hat{\boldsymbol{\beta}}$  with robust variance estimate  $\hat{\mathbf{V}}$ . Second, make a draw  $\boldsymbol{\beta}^*$  from the  $N(\hat{\boldsymbol{\beta}}, \hat{\mathbf{V}})$  distribution, and then for each observation  $i$  of the  $N_{\text{miss}}$  observations missing data on  $x$ , make a draw from the Poisson distribution with mean  $\exp(\mathbf{z}'_{mi}\hat{\boldsymbol{\beta}})$ , where  $\mathbf{z}_{mi}$  denotes the  $i$ th entry in  $\mathbf{Z}_m$ .

This yields one set of  $N_{\text{miss}}$  imputed values on  $\mathbf{x}_m$ . To obtain additional sets of imputed values, repeat the second step.

This imputation method can be given a Bayesian interpretation. Then  $\beta^*$  is a draw from  $N(\hat{\beta}, \hat{\mathbf{V}})$ , which is the asymptotic approximation to the posterior distribution of  $\beta$  given the noninformative prior  $\pi(\beta) \propto \text{constant}$ .

This method can be adapted to impute missing values for a single variable by using other parametric models, such as logit for binary data or normal with independent and identically distributed errors for continuous data. To make parametric assumptions more reasonable, the imputation might be applied to a transformation of  $x$ . For example, if  $x$  is right skewed, then first impute  $\ln x$ , and then set  $x = \exp(\ln x)$ .

Note that if the variable  $x$  with missing data is to be ultimately included in a regression with completely observed dependent variable  $y$ , then the variables  $\mathbf{z}$  used in imputing  $x$  should include  $y$ . Variables  $x$  and  $y$  are related, and we have made the assumption of MAR.

### 30.5.4 Imputation by data augmentation

We now consider simultaneously imputing several variables with missing values, under the assumption of joint normality of all variables. The imputation method is a Bayesian method that uses data augmentation to treat the missing values as parameters and then imputes these missing values as MCMC draws from the posterior.

Partition  $\mathbf{w}$  into  $(\mathbf{x}, \mathbf{z})$ , where  $\mathbf{x}$  denotes one or more incompletely observed variables and  $\mathbf{z}$  denotes completely observed variables to be used in imputation. Suppose that  $\mathbf{x}_i \sim N(\Pi \mathbf{z}_i, \Sigma)$ , where  $\Pi$  is a matrix of unknown coefficients and  $\Sigma$  is an unknown variance–covariance matrix. Because normality is assumed, it may be necessary to first transform some variables, such as taking the natural logarithm for right-skewed positive data.

For the incompletely observed variables  $\mathbf{x}$ , consider the partition  $\mathbf{X} = (\mathbf{X}_o, \mathbf{X}_m)$ , where  $\mathbf{X}_o$  contains observed values and  $\mathbf{X}_m$  contains

missing values. The goal is to obtain draws of the missing values  $\mathbf{X}_m$  given the observed data  $\mathbf{X}_o$  and  $\mathbf{Z}$ . Note that here all observations on  $\mathbf{z}$  are used, whereas the preceding regression-based imputation used only  $\mathbf{Z}_o$ .

We apply the data-augmentation method of section [30.4.1](#), treating  $\mathbf{X}_m$  as additional parameters. The draws from the model's posterior density  $p(\boldsymbol{\Pi}, \boldsymbol{\Sigma}, \mathbf{X}_m | \mathbf{Z}, \mathbf{X}_o)$  can be obtained using the Gibbs sampler, with alternating draws from the conditional posterior  $p(\boldsymbol{\Pi}, \boldsymbol{\Sigma} | \mathbf{X}_m, \mathbf{Z}, \mathbf{X}_o)$ , called the P step (posterior step), and the conditional posterior  $p(\mathbf{X}_m | \boldsymbol{\Pi}, \boldsymbol{\Sigma}, \mathbf{Z}, \mathbf{X}_o)$ , called the I step (imputation step). The conditional posteriors depend on the distribution of the data and the prior distributions of the parameters.

As already noted, we assume that  $\mathbf{x}_i \sim N(\boldsymbol{\Pi}\mathbf{z}_i, \boldsymbol{\Sigma})$ . A uniform prior is assumed for  $\boldsymbol{\Pi}$ , and an inverted Wishart prior is assumed for  $\boldsymbol{\Sigma}$ . The Wishart prior depends on two parameters, and the Stata default settings for these parameters lead to prior  $\pi(\boldsymbol{\Pi}, \boldsymbol{\Sigma}) \propto \boldsymbol{\Sigma}^{-(p+1)/2}$ , where  $p = \text{dim}(\mathbf{x})$ .

The data-augmentation method in the preceding section's probit example was used to obtain posterior draws of the parameters, so the posterior draws of the augmented latent variables were discarded. For multiple imputation, we instead discard the posterior draws of the parameters and keep the posterior draws of the augmented variables  $\mathbf{X}_m^*$ . To ensure that the imputed datasets are independent, we follow the default, which is to use every hundredth posterior draw.

### 30.5.5 The `mi import`, `mi impute`, and `mi estimate` commands

In some cases, multiply imputed datasets are already available. If these data are available as Stata `mi` data, then the data can be read in using the `use` command. If these multiply imputed data are in other formats, then some formats can be read in using the `mi import` command.

To instead impute data oneself, one uses the key command `mi impute`, which has format

```
mi impute method ... [, options]
```

The *method* for regression-based imputation of a single variable includes `regress` for continuous data, `intreg` for interval recorded continuous data, `truncreg` for truncated normal data, `pmm` for predictive mean matching of a continuous variable, `poisson` and `nbreg` for count data, `logit` for binary data, `mlogit` for unordered multinomial data, and `ologit` for ordered multinomial data. *method mvn* implements data augmentation imputation of one or more variables based on a multivariate normal model, *method chained* implements multivariate imputation of different types of variables by using a sequence of univariate full-conditional specifications, and *usermethod* allows you to add your own imputation methods.

Given multiple-imputation datasets, obtained either by use of the `mi impute` command or from an external source, the actual econometric model of interest is fit using the `mi estimate` command, which has the format

```
mi estimate [, options] : estimation_command ...
```

where *options* specific to `mi estimate` include `nimputations(#)`, the number of imputations.

Other `mi` declaration commands need to be used ahead of use of the `mi impute` and `mi estimate` commands. These are illustrated in the example below.

## 30.6 Multiple-imputation example

We present in detail a simple example where data on a single regressor are MAR.

We create a dataset of 10,000 observations with  $y = 0 + x_2 + 2x_3 + \varepsilon$ , where  $x_2 \sim N(0, 1)$ ,  $x_3 \sim N(0, 1)$ ,  $\text{Cor}(x_2, x_3) = 0.5$ , and  $\varepsilon \sim N(0, 1)$ .

```
. * Create complete data with regressors x2 and x3 correlated
. clear
. set seed 10101
. matrix C = (1, .5 \ .5, 1)
. drawnorm x2 x3, n(10000) corr(C)
(obs 10,000)
. generate y = 0 + 1*x2 + 2*x3 + rnormal(0,1)
. summarize
```

| Variable | Obs    | Mean      | Std. dev. | Min       | Max      |
|----------|--------|-----------|-----------|-----------|----------|
| x2       | 10,000 | .0012519  | 1.00765   | -4.119125 | 3.596714 |
| x3       | 10,000 | -.0073642 | 1.009584  | -3.410766 | 3.781023 |
| y        | 10,000 | -.0254453 | 2.86018   | -10.65987 | 11.10632 |

Ordinary least-squares (OLS) estimation on all 10,000 observations yields

```
. * OLS on complete data
. regress y x2 x3
```

| Source   | SS         | df    | MS         | Number of obs | = | 10,000   |
|----------|------------|-------|------------|---------------|---|----------|
| Model    | 71948.605  | 2     | 35974.3025 | F(2, 9997)    | = | 36513.04 |
| Residual | 9849.49785 | 9,997 | .985245359 | Prob > F      | = | 0.0000   |
| Total    | 81798.1028 | 9,999 | 8.18062835 | R-squared     | = | 0.8796   |
|          |            |       |            | Adj R-squared | = | 0.8796   |
|          |            |       |            | Root MSE      | = | .9926    |

| y     | Coefficient | Std. err. | t      | P> t  | [95% conf. interval] |
|-------|-------------|-----------|--------|-------|----------------------|
| x2    | 1.023646    | .0114128  | 89.69  | 0.000 | 1.001275 1.046018    |
| x3    | 1.990495    | .0113909  | 174.74 | 0.000 | 1.968167 2.012824    |
| _cons | -.0120684   | .0099264  | -1.22  | 0.224 | -.0315261 .0073893   |

The 95% confidence intervals for the intercept and  $x_3$  include their DGP values. The 95% confidence interval for  $x_2$  does not quite include the DGP

value of 1.0, but this is not of concern because this will happen in 5% of simulations.

Now suppose some observations on  $x_2$  are missing, and these are missing when  $x_3$  takes high values, specifically, if  $x_3 > u$ , where  $u \sim N(0.5, 0.5^2)$ .

```
. * Create incomplete data by dropping some x2 (MAR)
. set seed 12345
. replace x2 = . if x3 > rnormal(0.5, 0.5)
(3,282 real changes made, 3,282 to missing)
. summarize
Variable Obs Mean Std. dev. Min Max
x2 6,718 -.2407734 .9585193 -4.119125 3.355527
x3 10,000 -.0073642 1.009584 -3.410766 3.781023
y 10,000 -.0254453 2.86018 -10.65987 11.10632
. qui save incomplete, replace
```

We lose approximately 33% of the observations on  $x_2$ .

Because missingness of  $x_2$  depends only on  $x_3$  and not on  $y$ , which depends in part on the unobservable  $\varepsilon$ , for analysis of  $(y, x_2, x_3)$ , the data are MAR. So OLS regression of  $y$  on  $x_2$  and  $x_3$  with incomplete observations dropped will still provide consistent estimates. We obtain

```
. * OLS on incomplete data
. regress y x2 x3
Source SS df MS Number of obs = 6,718
 F(2, 6715) = 15243.03
Model 30330.9962 2 15165.4981 Prob > F = 0.0000
Residual 6680.84331 6,715 .994913374 R-squared = 0.8195
Total 37011.8395 6,717 5.51017411 Adj R-squared = 0.8194
 Root MSE = .99745

```

|       | y         | Coefficient | Std. err. | t     | P> t      | [95% conf. interval] |
|-------|-----------|-------------|-----------|-------|-----------|----------------------|
| x2    | 1.034929  | .013885     | 74.54     | 0.000 | 1.00771   | 1.062148             |
| x3    | 1.991878  | .0174389    | 114.22    | 0.000 | 1.957692  | 2.026064             |
| _cons | -.0060802 | .0144895    | -0.42     | 0.675 | -.0344842 | .0223238             |

The fitted coefficients are again essentially equal to their DGP values; the 95% confidence interval for the coefficient of  $x_2$  does not quite contain 1.0,

as was the case for the complete data. Compared with the complete data, the standard error of  $x_2$  has increased from 0.0114 to 0.0139, and the standard error of  $x_3$  has increased from 0.0114 to 0.0174.

If  $x_2$  was MCAR, there would be no benefit in imputing the missing observations. However, the missingness of  $x_2$  was related to values of  $x_3$ , and  $x_2$  and  $x_3$  are correlated. So there is the possibility of imputing the missing values  $x_2$ , using the observed data on the other variables, and improving efficiency of subsequent estimation.

### 30.6.1 Describing missing-value patterns

For regular datasets, the `misstable` commands can be used to summarize the missing patterns in the data. For the current data, we obtain

```
. * Summarize missingness
. misstable summarize
```

| Variable | Obs=. | Obs>. | Obs<. | Obs<.         |           |          |
|----------|-------|-------|-------|---------------|-----------|----------|
|          |       |       |       | Unique values | Min       | Max      |
| x2       | 3,282 |       | 6,718 | >500          | -4.119125 | 3.355527 |

```
. misstable patterns
```

Missing-value patterns  
(1 means complete)

| Percent | Pattern |   |
|---------|---------|---|
|         | 1       | 0 |
| 67%     | 1       |   |
| 33      | 0       |   |
| 100%    |         |   |

Variables are (1) x2

Data are missing for 3,282 values of  $x_2$  and are available for 6,718 (a total of 10,000). Thus, 33% are missing and 67% are available.

Note that if the dataset is already in Stata `mi` dataset form, then we need to instead use the `mi misstable` commands. The default for these commands is to give the missing-data patterns for only the original (nonimputed) data.

### 30.6.2 Data imputation

There are choices to make in imputing the data. First, choosing which variables will be used to impute  $x_2$ . We impute using data on both  $x_3$  and  $y$ .

Second, choosing which imputation method is used. Because only a single variable is missing, we could use the `mi impute regress x2 y x3` command. This assumes that the conditional distribution of  $x_2$  given  $y$  and  $x_3$  is normal. We instead use the more general `mi impute mvn x2 = y x3` command, which can also be used if more than one variable is missing. This command assumes that the data  $(x_2, y, x_3)$  are joint normally distributed, a reasonable assumption here, but in other applications, it may be necessary to first transform some of the variables.

Before running the multiple imputation, we first use the `mi set mlong` command to say that the imputed dataset will be stored in the `mlong` data style and use the `mi register` command to declare which variables have missing observations and which have complete observations.

We then impute only three times to reduce the length of output in this illustrative example. We manually set the burn-in at 100 draws, which is hopefully enough for the chain to have converged, and keep every 100th draw, which hopefully is a big enough separation to ensure that the imputed values are independent. (These are the Stata defaults, so their explicit use here is for expository purposes.) We obtain

```

. * Declare dataset type (long), register imputed variables, perform 3 imputations
. mi set mlong
. mi register imputed x2
(3282 m=0 obs now marked as incomplete)
. mi register regular y x3
. mi impute mvn x2 = y x3, add(3) rseed(10101) burnin(100) burnbetween(100)
Performing EM optimization:
note: 3282 observations omitted from EM estimation because of all imputation
 variables missing.
 observed log likelihood = -448.04782 at iteration 1
Performing MCMC data augmentation ...
Multivariate imputation Imputations = 3
Multivariate normal regression added = 3
Imputed: m=1 through m=3 updated = 0
Prior: uniform Iterations = 300
 burn-in = 100
 between = 100

```

| Variable | Observations per m |            |         | Total |
|----------|--------------------|------------|---------|-------|
|          | Complete           | Incomplete | Imputed |       |
| x2       | 6718               | 3282       | 3282    | 10000 |

(Complete + Incomplete = Total; Imputed is the minimum across  $m$  of the number of filled-in observations.)

The default uniform prior is used.

The following data and variables are created.

```

. * Multiple imputation creates the following data and variables
. summarize

```

| Variable | Obs    | Mean     | Std. dev. | Min       | Max      |
|----------|--------|----------|-----------|-----------|----------|
| x2       | 16,564 | .1851142 | .9932464  | -4.119125 | 3.747372 |
| x3       | 19,846 | .4858939 | .9898968  | -3.410766 | 3.781023 |
| y        | 19,846 | 1.205346 | 2.808125  | -10.65987 | 11.10632 |
| _mi_m    | 19,846 | .9922402 | 1.153583  | 0         | 3        |
| _mi_id   | 19,846 | 4980.141 | 2888.991  | 1         | 10000    |
| _mi_miss | 10,000 | .3282    | .4695815  | 0         | 1        |

Note that rather than save three datasets each with 30,000 observations, we save storage space by storing nonmissing data only once. This leads to the

creation of the indicator variables `_mi_m`, `_mi_id`, and `_mi_miss` that appear in every `mi` dataset.

The indicator variable `_mi_m` equals 0 for the original 10,000 observations, 1 for the first set of 3,282 imputed observations, 2 for the second set of 3,282 imputed observations, and 3 for the third set of 3,282 imputed observations. In all, there are  $10000 + 3 \times 3282 = 19846$  observations. The variable `x2` was missing for 3,282 observations, so summary statistics report 16,564 ( $= 19846 - 3282$ ) observed values for `x2`.

Variable `_mi_id` takes value 1 to 10,000 for the original 10,000 observations and thereafter for imputed data equals the `_mi_id` in the original data, so `_mi_id` gives the identifier for the imputed observations.

The indicator variable `_mi_miss` takes value 1 if any data are missing in the first 10,000 observations and value 0 if data are not missing in the first 10,000 observations and takes the missing value (.) for the subsequent imputed observations.

As for any MCMC method, there is no guarantee that the chain has converged. It is more likely the longer the chain runs, so we might increase the `burnin()`. The Stata manual entry [MI] **mi impute mvn** demonstrates use of diagnostics for convergence of the parameters using results stored in the option `savewlf()`.

An obvious check of both the imputation model and chain convergence is to see whether the imputed values of missing observations are sensible. The `mi xeq` prefix can be used to apply Stata data-descriptive commands such as `summarize` or `tabulate` or `kdensity` to one or more of the original and imputed datasets.

We summarize the data for the original data as well as the first and third imputed datasets.

```

. * Check reasonableness of imputed x2
. mi xeq 0 1 3: summarize y x2 x3
m=0 data:
-> summarize y x2 x3

```

| Variable | Obs    | Mean      | Std. dev. | Min       | Max      |
|----------|--------|-----------|-----------|-----------|----------|
| y        | 10,000 | -.0254453 | 2.86018   | -10.65987 | 11.10632 |
| x2       | 6,718  | -.2407734 | .9585193  | -4.119125 | 3.355527 |
| x3       | 10,000 | -.0073642 | 1.009584  | -3.410766 | 3.781023 |

```

m=1 data:
-> summarize y x2 x3

```

| Variable | Obs    | Mean      | Std. dev. | Min       | Max      |
|----------|--------|-----------|-----------|-----------|----------|
| y        | 10,000 | -.0254453 | 2.86018   | -10.65987 | 11.10632 |
| x2       | 10,000 | -.0050421 | 1.00424   | -4.119125 | 3.586417 |
| x3       | 10,000 | -.0073642 | 1.009584  | -3.410766 | 3.781023 |

```

m=3 data:
-> summarize y x2 x3

```

| Variable | Obs    | Mean      | Std. dev. | Min       | Max      |
|----------|--------|-----------|-----------|-----------|----------|
| y        | 10,000 | -.0254453 | 2.86018   | -10.65987 | 11.10632 |
| x2       | 10,000 | -.0045282 | 1.00216   | -4.119125 | 3.747372 |
| x3       | 10,000 | -.0073642 | 1.009584  | -3.410766 | 3.781023 |

As expected, the completely observed variables `y` and `x3` take the same values in all datasets. The incompletely observed variable `x2` takes a similar range of values in the first and third imputed datasets, with mean close to 0 and standard deviation close to 1.

The mean of `x2` in the completed datasets of 10,000 observations is about 0.25 standard deviations higher than the mean of `x2` in the observed dataset of 6,718 observations. In this example, we know the DGP and so expect this difference because the missing values of `x2` will generally be higher values because `x2` is missing if `x3` takes higher values and `x2` and `x3` are positively correlated. Without this knowledge, further investigation might be warranted.

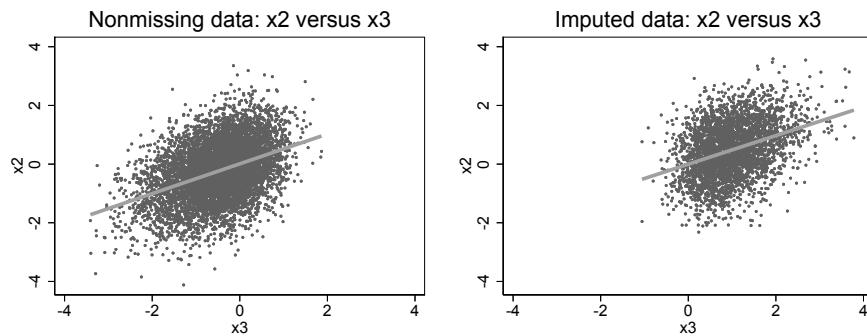
The following commands graph the relationship between `x2` and `x3` for the original 6,718 completely observed observations and for the first set of 3,282 imputed values.

```

. * Plot x2 against x3 for nonmissing data and first imputation
. qui graph twoway (scatter x2 x3 if _mi_m==0, msize(tiny))
> (lfit x2 x3 if _mi_m==0, lwidth(thick)), ytitle("x2")
> title("Nonmissing data: x2 versus x3") legend(off) saving(graph1, replace)
. qui graph twoway (scatter x2 x3 if _mi_m==1, msize(tiny))
> (lfit x2 x3 if _mi_m==1, lwidth(thick)), ytitle("x2")
> title("Imputed data: x2 versus x3") legend(off) saving(graph2, replace)
. graph combine graph1.gph graph2.gph, ycommon xcommon iscale(1.2)
> ysize(2.5) xsize(6) rows(1)

```

Figure 30.3 presents the results. The relationship between  $x_2$  and  $x_3$  is similar across the two datasets, with the imputed dataset values shifted to the right with higher values of  $x_2$  and  $x_3$ .



**Figure 30.3.** A basic scatterplot of log earnings on hours

### 30.6.3 Estimation using imputed data

Before fitting the model using imputed data, we describe the imputed data.

```

. * Describe the imputed data
. mi describe
Style: mlong
Observations:
 Complete 6,718
 Incomplete 3,282 (M = 3 imputations)

Total 10,000

Variables:
 Imputed: 1; x2(3282)
 Passive: 0
 Regular: 2; y x3
 System: 3; _mi_m _mi_id _mi_miss
 (there are no unregistered variables)

```

The original data had 10,000 observations of which 6,718 were complete. Here 3 sets of imputed values for the 3,282 incomplete observations are obtained. We use the `mi estimate` command to perform OLS estimation in each of the 3 complete datasets, each with 6,718 complete observations and 3,282 imputed observations (with imputation here only of `x2`).

```

. * Fit the model
. mi estimate, dots: regress y x2 x3
Imputations (3):
... done

Multiple-imputation estimates
Linear regression
DF adjustment: Small sample
Model F test: Equal FMI
Within VCE type: OLS
Implications
Number of obs = 10,000
Average RVI = 0.3207
Largest FMI = 0.5336
Complete DF = 9997
DF: min = 9.93
 avg = 1,369.10
 max = 2,426.18
F(2, 29.6) = 24673.11
Prob > F = 0.0000

```

| y     | Coefficient | Std. err. | t      | P> t  | [95% conf. interval] |
|-------|-------------|-----------|--------|-------|----------------------|
| x2    | 1.02051     | .0154657  | 65.99  | 0.000 | .9860178 1.055001    |
| x3    | 2.003095    | .0115281  | 173.76 | 0.000 | 1.980489 2.025701    |
| _cons | -.0049432   | .010129   | -0.49  | 0.626 | -.0248102 .0149237   |

The estimated coefficients are close to their DGP values of, respectively, 1, 2, and 0. Compared with regression using the incomplete data, the standard

error of  $x_2$  has increased from 0.0139 to 0.0155, and the standard error of  $x_3$  has decreased from 0.0174 to 0.0115. The increase in the standard error of  $x_2$  is due to increased variability due to few imputations. Below, we increase the number of imputations to 10 and find that all the standard errors have decreased.

Inference is based on  $t$  statistics and  $F$  statistics with degrees of freedom that depend in a complicated way on the number of observations, the consequences of missing data, and the number of imputations.

The `mi convert` command can be used to create separate datasets on the original observations and for each of the imputations. These could be exported and used by other analysts.

For the current example, we have

```
. * Create three complete datasets with imputed
. mi convert flongsep midata, replace clear
(files midata.dta _1_midata.dta _2_midata.dta _3_midata.dta created)
```

Each of these imputed datasets is composed of nonmissing data on all variables plus 3,228 imputed observations for  $x_2$ .

The following code demonstrates that running three separate regressions on these three imputed datasets and averaging the resulted three estimated coefficients leads to the coefficient estimate reported in the preceding `mi estimate` command. We do so for the coefficient of  $x_2$ .

```
. * Perform OLS estimation on each of these three imputed datasets
. use _1_midata, clear
. qui regress y x2 x3, noheader
. scalar b1x2 = _b[x2]
. use _2_midata, clear
. qui regress y x2 x3, noheader
. qui scalar b2x2 = _b[x2]
. use _3_midata, clear
. qui regress y x2 x3, noheader
. scalar b3x2 = _b[x2]
. display "Ave OLS coeff of x2 over 3 imputed samples = " (b1x2 + b2x2 + b3x2)/3
Ave OLS coeff of x2 over 3 imputed samples = 1.0205095
```

The average of the 3 coefficients of  $x_2$  is 1.02051, as expected.

We conclude by repeating the analysis using 10 imputations.

```
. * Impute and estimate with 10 imputations
. use incomplete, clear
. mi set mlong
. mi register imputed x2
(3282 m=0 obs now marked as incomplete)
. mi register regular y x3
. qui mi impute mvn x2 = y x3, add(10) rseed(10101) burnin(100) burnbetween(100)
. mi estimate, dots: regress y x2 x3
Imputations (10):
.....10 done

Multiple-imputation estimates
Linear regression
DF adjustment: Small sample
Model F test: Equal FMI
Within VCE type: OLS

```

|               |     |          |
|---------------|-----|----------|
| Imputations   | =   | 10       |
| Number of obs | =   | 10,000   |
| Average RVI   | =   | 0.7354   |
| Largest FMI   | =   | 0.5129   |
| Complete DF   | =   | 9997     |
| DF:           | min | = 37.55  |
|               | avg | = 69.79  |
|               | max | = 117.71 |
| F( 2, 68.6)   | =   | 20513.58 |
| Prob > F      | =   | 0.0000   |

| y     | Coefficient | Std. err. | t      | P> t  | [95% conf. interval] |
|-------|-------------|-----------|--------|-------|----------------------|
| x2    | 1.018753    | .0134219  | 75.90  | 0.000 | .9921737 1.045333    |
| x3    | 1.993356    | .0159557  | 124.93 | 0.000 | 1.961043 2.025669    |
| _cons | -.0127999   | .012918   | -0.99  | 0.326 | -.0386979 .0130981   |

Compared with regression using the incomplete data, the standard error of  $x_2$  has decreased from 0.0139 to 0.0134, and the standard error of  $x_3$  has decreased from 0.0174 to 0.0160.

## 30.7 Additional resources

The [BAYES] *Stata Bayesian Analysis Reference Manual* provides a very detailed exposition of Bayesian analysis and MCMC computation. The key commands are the `bayes` prefix and the `bayesmh` command. Bayesian analysis is presented in many complete books, including [Koop \(2003\)](#) and [Gelman et al. \(2013\)](#), and in [Cameron and Trivedi \(2005, chap. 13\)](#).

The [MI] *Stata Multiple-Imputation Reference Manual* covers the multiple-imputation commands; see also [Cameron and Trivedi \(2005, chap. 27\)](#).

## 30.8 Exercises

1. This question adapts the program in section [30.3](#) from a probit model to a logit model. Use the same DGP and seed as in section [30.2.1](#), except replace the line `generate ystar = 0.5*x+rnormal(0,1)` with the three lines `gen u=runiform()` and `gen ulogistic=ln(u)-ln(1-u)` and `gen ystar=0.5*x+ulogistic`. Fit by the usual logit MLE, and report the results. Fit the logit model with flat priors for the intercept and slope parameters using Stata command `bayesmh` and default settings. For the slope coefficient  $\beta_2$ , give the posterior mean, standard deviation, and a 95% Bayesian posterior credible interval. Compare the posterior mean and standard deviation of  $\beta_2$  with the logit MLE of  $\hat{\beta}_2$  and its standard error. Comment on any differences between the two. Is this what you expect?
2. Continue the example of question 1. Adapt the Mata code to run MCMC under MH for the logit model. Run this code. Note that the Mata command `invlogit(A)` computes  $\Lambda(a_{ij})$  for each element of the matrix  $A$ . Compare your results with those obtained using the `bayesmh` command. Does your MH method appear to work well? Is there much serial correlation between draws? Is the acceptance rate high? Explain your answer.
3. This question adapts the program in section [30.4](#) from a probit model to a tobit model. For simplicity,  $\sigma^2$  is known with  $\sigma^2 = 1$ , so we need obtain only the posterior for  $\beta$ . Use the same DGP and seed as in section [30.2.1](#), except replace the line `generate y=(ystar>0)` with `gen y=ystar` and `replace y=0 if ystar<0`. Fit the tobit model by the usual logit MLE, and report the results. Adapt the Mata code for the Gibbs sampler and data augmentation for the probit model to the tobit model (with known  $\sigma^2 = 1$ ). Run this code. Compare your results with the ML estimates. Does your MCMC method appear to work well? Is there much serial correlation between draws? Is the acceptance rate high? Explain your answer.
4. This question adapts the multiple-imputation example. Use the command `mi impute regress` with seed 10101, 10 imputations, and  $x_2$  imputed using data on  $y$  and  $x_3$ . Summarize the imputed datasets, and

regress  $y$  on  $x_2$  and  $x_3$  using the imputed data. Compare your results with those in section [30.6](#) based on the `mi impute mvn` command.

5. Generate data using the following commands: `clear` and `set seed 10101` and `matrix C=(1,.5/5,1)` and `drawnorm x2 x3, n(10000) corr(C)` and, finally, `gen y=0+1*x2+2*x3+rnormal(0,1)`. Summarize the DGP. Estimate by OLS. Now, give commands `set seed 12345` and `replace x2=. if x3>0.5`. Is the missing-data mechanism MCAR, MAR, or MNAR? For each of the following methods, obtain the OLS coefficient estimates and state whether you believe slope coefficients are consistently estimated and standard errors are consistently estimated: i) case deletion; ii) mean imputation for the missing variable; iii) mean imputation for the missing variable plus a dummy variable for missing; iv) imputation by regressing the fitted value of the regressor on other completely observed regressors and using the fitted values from this regression to impute the missing regressor; v) multiple imputation using the `mi impute regress` command with seed 10101, 10 imputations, and  $x_2$  imputed using data on  $y$  and  $x_3$ . Which methods give valid results?

# 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. 2003. Semiparametric instrumental variable estimation of treatment response models. *Journal of Econometrics* 113: 231–263. [https://doi.org/10.1016/S0304-4076\(02\)00201-4](https://doi.org/10.1016/S0304-4076(02)00201-4).
- \_\_\_\_\_. 2021. Using synthetic controls: Feasibility, data requirements, and methodological aspects. *Journal of Economic Literature* 59: 391–425. <https://doi.org/10.1257/jel.20191450>.
- Abadie, A., J. D. Angrist, and G. W. Imbens. 2002. Instrumental variables estimates of the effect of subsidized training on the quantiles of trainee earnings. *Econometrica* 70: 91–117. <https://doi.org/10.1111/1468-0262.00270>.
- Abadie, A., S. Athey, G. W. Imbens, and J. M. Wooldridge. 2020. Sampling-based versus design-based uncertainty in regression analysis. *Econometrica* 88: 265–296. <https://doi.org/10.3982/ECTA12675>.
- \_\_\_\_\_. 2022. When should you adjust standard errors for clustering? ArXiv Working Paper No. arXiv:1710.02926. <https://doi.org/10.48550/arXiv.1710.02926>.
- Abadie, A., and M. D. Cattaneo. 2019. Econometric methods for program evaluation. *Annual Review of Economics* 10: 465–503. <https://doi.org/10.1146/annurev-economics-080217-053402>.
- Abadie, A., A. Diamond, and J. Hainmueller. 2010. Synthetic control methods for comparative case studies: Estimating the effect of California’s tobacco control program. *Journal of the American Statistical Association* 105: 493–505. <https://doi.org/10.1198/jasa.2009.ap08746>.
- \_\_\_\_\_. 2014. synth: Stata module to implement synthetic control methods for comparative case studies. Statistical Software Components S457334, Department of Economics, Boston College. <https://ideas.repec.org/c/boc/bocode/s457334.html>.

- Abadie, A., and G. W. Imbens. 2006. Large sample properties of matching estimators for average treatment effects. *Econometrica* 74: 235–267. <https://doi.org/10.1111/j.1468-0262.2006.00655.x>.
- Abrevaya, J., and S. G. Donald. 2017. A GMM approach for dealing with missing data on regressors. *Review of Economics and Statistics* 99: 657–662. [https://doi.org/10.1162/REST\\_a\\_00645](https://doi.org/10.1162/REST_a_00645).
- Acemoglu, D., S. Johnson, and J. A. Robinson. 2001. The colonial origins of comparative development: An empirical investigation. *American Economic Review* 91: 1369–1401.  
<https://doi.org/10.1257/aer.91.5.1369>.
- Ahrens, A., C. B. Hansen, and M. E. Schaffer. 2018. pdslasso: Stata module for post-selection and post-regularization OLS or IV estimation and inference. Statistical Software Components S458459, Department of Economics, Boston College.  
<https://ideas.repec.org/c/boc/bocode/s458459.html>.
- \_\_\_\_\_. 2019. lassopack: Model selection and prediction with regularized regression in Stata. ArXiv Working Paper No. arXiv:1901.05397. <https://doi.org/10.48550/arXiv.1901.05397>.
- Amemiya, T. 1981. Qualitative response models: A survey. *Journal of Economic Literature* 19: 1483–1536.
- Andresen, M. E. 2018. Exploring marginal treatment effects: Flexible estimation using Stata. *Stata Journal* 18: 118–158.  
<https://doi.org/10.1177/1536867X1801800108>.
- Angrist, J. D., and G. W. Imbens. 2005. Two-stage least squares estimation of average causal effects in models with variable treatment intensity. *Journal of the American Statistical Association* 90: 431–442. <https://doi.org/10.1080/01621459.1995.10476535>.
- Angrist, J. D., G. W. Imbens, and D. B. Rubin. 1996. Identification of causal effects using instrumental variables. *Journal of the American Statistical Association* 91: 444–455.  
<https://doi.org/10.1080/01621459.1996.10476902>.

- Angrist, J. D., and J.-S. Pischke. 2009. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton, NJ: Princeton University Press.
- Anselin, L. 1988. *Spatial Econometrics: Methods and Models*. Dordrecht: Kluwer.
- Arellano, M. 2003. *Panel Data Econometrics*. New York: Oxford University Press.
- Athey, S., and G. W. Imbens. 2006. Identification and inference in nonlinear difference-in-differences models. *Econometrica* 74: 431–497. <https://doi.org/10.1111/j.1468-0262.2006.00668.x>.
- \_\_\_\_\_. 2019. Machine learning methods economists should know about. *Annual Review of Economics* 11: 685–725. <https://doi.org/10.1146/annurev-economics-080217-053433>.
- Baicker, K., S. L. Taubman, H. L. Allen, M. Bernstein, J. H. Gruber, J. P. Newhouse, E. C. Schneider, B. J. Wright, A. M. Zaslavsky, and A. N. Finkelstein. 2013. The Oregon experiment—Effects of Medicaid on clinical outcomes. *New England Journal of Medicine* 368: 1713–1722. <https://doi.org/10.1056/NEJMsa1212321>.
- Baker, M. 2016. qregpd: Stata module to perform quantile regression for panel data. Statistical Software Components S458157, Department of Economics, Boston College. <https://ideas.repec.org/c/boc/bocode/s458157.html>.
- Baltagi, B. H. 2021. *Econometric Analysis of Panel Data*. 6th ed. Cham, Switzerland: Springer.
- Baltagi, B. H., and S. H. Song. 2006. Unbalanced panel data: A survey. *Statistical Papers* 47: 493–523. <https://doi.org/10.1007/s00362-006-0304-0>.
- Banerjee, A. V., and E. Duflo. 2011. *Poor Economics: A Radical Rethinking of the Way to Fight Global Poverty*. New York: Public Affairs.
- Barker, M. 2014. sls—Semiparametric least squares from Ichimura, 1993. GitHub. <https://github.com/michaelbarker/stata-sls>.

- Barron, M. 2014. loocv: Stata module to perform leave-one-out cross-validation. Statistical Software Components S457926, Department of Economics, Boston College.  
<https://ideas.repec.org/c/boc/bocode/s457926.html>.
- Bartolucci, F. 2015. cquad: Stata module to perform conditional maximum likelihood estimation of quadratic exponential models. Statistical Software Components S457891, Department of Economics, Boston College. <https://ideas.repec.org/c/boc/bocode/s457891.html>.
- Bartolucci, F., and V. Nigro. 2010. A dynamic model for binary panel data with unobserved heterogeneity admitting a  $\sqrt{n}$ -consistent conditional estimator. *Econometrica* 78: 719–733.  
<https://doi.org/10.3982/ECTA7531>.
- Batistatou, E., C. Roberts, and S. Roberts. 2014. Sample size and power calculations for trials and quasi-experimental studies with clustering. *Stata Journal* 14: 159–175.  
<https://doi.org/10.1177/1536867X1401400111>.
- Baum, C. F. 2016. *An Introduction to Stata Programming*. 2nd ed. College Station, TX: Stata Press.
- Becker, S. O., and A. Ichino. 2002. Estimation of average treatment effects based on propensity scores. *Stata Journal* 2: 358–377.  
<https://doi.org/10.1177/1536867X0200200403>.
- Belloni, A., D. Chen, V. Chernozhukov, and C. B. Hansen. 2012. Sparse models and methods for optimal instruments with an application to eminent domain. *Econometrica* 80: 2369–2429.  
<https://doi.org/10.3982/ECTA9626>.
- Belloni, A., and V. Chernozhukov. 2013. Least squares after model selection in high-dimensional sparse models. *Bernoulli* 19: 521–547.  
<https://doi.org/10.3150/11-BEJ410>.
- Belloni, A., V. Chernozhukov, and C. B. Hansen. 2014. Inference on treatment effects after selection among high-dimensional controls. *Review of Economic Studies* 81: 608–650.  
<https://doi.org/10.1093/restud/rdt044>.

- Belloni, A., V. Chernozhukov, and Y. Wei. 2016. Post-selection inference for generalized linear models with many controls. *Journal of Business and Economics Statistics* 34: 606–619.  
<https://doi.org/10.1080/07350015.2016.1166116>.
- Berry, S., J. Levinsohn, and A. Pakes. 1995. Automobile prices in market equilibrium. *Econometrica* 63: 841–890.  
<https://doi.org/10.2307/2171802>.
- Blundell, R., and J. L. Powell. 2003. Endogeneity in nonparametric and semiparametric regression models. In Vol. 2 of *Advances in Economics and Econometrics: Theory and Applications, Eight World Congress*, ed. M. Dewatripont, L. P. Hansen, and S. J. Turnovsky, 312–357. Cambridge: Cambridge University Press.  
<https://doi.org/10.1017/CBO9780511610257.011>.
- Bollen, K. A. 1989. *Structural Equations with Latent Variables*. New York: Wiley.
- Borah, B. J., and A. Basu. 2013. Highlighting differences between conditional and unconditional quantile regression approaches through an application to assess medication adherence. *Health Economics* 22: 1052–1070. <https://doi.org/10.1002/hec.2927>.
- Borgen, N. T. 2016. Fixed effects in unconditional quantile regression. *Stata Journal* 16: 403–415.  
<https://doi.org/10.1177/1536867X1601600208>.
- Borusyak, K., X. Jaravel, and J. Spiess. 2022. Revisiting event study designs: Robust and efficient estimation. ArXiv Working Paper No. arXiv:2108.12419. <https://doi.org/10.48550/arXiv.2108.12419>.
- Callaway, B., and P. H. C. Sant'Anna. 2021. Difference-in-Differences with multiple time periods. *Journal of Econometrics* 225: 200–230.  
<https://doi.org/10.1016/j.jeconom.2020.12.001>.
- Calonico, S., M. D. Cattaneo, M. H. Farrell, and R. Titiunik. 2017. rdrobust: Software for regression-discontinuity designs. *Stata Journal* 17: 372–404. <https://doi.org/10.1177/1536867X1701700208>.

- Calonico, S., M. D. Cattaneo, and R. Titiunik. 2014. Robust data-driven inference in the regression-discontinuity design. *Stata Journal* 14: 909–946. <https://doi.org/10.1177/1536867X1401400413>.
- \_\_\_\_\_. 2015. Optimal data-driven regression discontinuity plots. *Journal of the American Statistical Association* 110: 1753–1769. <https://doi.org/10.1080/01621459.2015.1017578>.
- 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.
- Canay, I. A. 2011. A simple approach to quantile regression for panel data. *Econometrics Journal* 14: 368–386. <https://doi.org/10.1111/j.1368-423X.2011.00349.x>.
- Card, D., D. S. Lee, Z. Pei, and A. Weber. 2015. Inference on causal effects in a generalized regression kink design. *Econometrica* 83: 2453–2483. <https://doi.org/10.3982/ECTA11224>.
- Carson, R. T., and Y. Sun. 2007. The tobit model with a non-zero threshold. *Econometrics Journal* 10: 488–502. <https://doi.org/10.1111/j.1368-423X.2007.00218.x>.
- Cattaneo, M. D. 2010. Efficient semiparametric estimation of multi-valued treatment effects under ignorability. *Journal of Econometrics* 155: 138–154. <https://doi.org/10.1016/j.jeconom.2009.09.023>.
- Cattaneo, M. D., D. M. Drukker, and A. D. Holland. 2013. Estimation of multivalued treatment effects under conditional independence. *Stata Journal* 13: 407–450. <https://doi.org/10.1177/1536867X1301300301>.
- Cerulli, G. 2014. ivtreatreg: A command for fitting binary treatment models with heterogeneous response to treatment and unobservable selection. *Stata Journal* 14: 453–480. <https://doi.org/10.1177/1536867X1401400301>.
- Cheng, T. C., and P. K. Trivedi. 2015. Attrition bias in panel data: A sheep in wolf's clothing? A case study based on the Mabel survey. *Health Economics* 24: 1101–1117. <https://doi.org/10.1002/hec.3206>.

- Chernozhukov, V., D. Chetverikov, M. Demirer, E. Duflo, C. B. Hansen, W. K. Newey, and J. Robins. 2018. Double/debiased machine learning for treatment and structural parameters. *Econometrics Journal* 21: C1–C68. <https://doi.org/10.1111/ectj.12097>.
- 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., I. Fernández-Val, and B. Melly. 2009. Inference on counterfactual distributions. ArXiv Working Paper No. arXiv:0904.0951. <https://doi.org/10.48550/arXiv.0904.0951>.
- \_\_\_\_\_. 2013. Inference on counterfactual distributions. *Econometrica* 81: 2205–2268. <https://doi.org/10.3982/ECTA10582>.
- Chernozhukov, V., and C. B. Hansen. 2006. Instrumental variable quantile regression inference for structural and treatment effect models. *Journal of Econometrics* 132: 491–525. <https://doi.org/10.1016/j.jeconom.2005.02.009>.
- \_\_\_\_\_. 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>.
- Chernozhukov, V., C. B. Hansen, and M. Spindler. 2015. Post-selection inference for generalized linear models with many controls. *American Economic Review* 105(5): 486–490. <https://doi.org/10.1257/aer.p20151022>.
- Chernozhukov, V., W. K. Newey, and R. Singh. 2022. Automatic debiased machine learning of causal and structural effects. *Econometrica* 90: 967–1027. <https://doi.org/10.3982/ECTA18515>.
- Chib, S. 2001. Markov chain Monte Carlo methods: Computation and inference. In Vol. 5 of *Handbook of Econometrics*, ed. J. J. Heckman and E. Leamer, 3569–3649. Amsterdam: North-Holland. [https://doi.org/10.1016/S1573-4412\(01\)05010-3](https://doi.org/10.1016/S1573-4412(01)05010-3).
- Cleves, M., W. W. Gould, and Y. V. Marchenko. 2016. *An Introduction to Survival Analysis Using Stata*. Rev. 3rd ed. College Station, TX: Stata

Press.

- Conley, T. G. 1999. GMM estimation with cross sectional dependence. *Journal of econometrics* 92: 1–45. [https://doi.org/10.1016/S0304-4076\(98\)00084-0](https://doi.org/10.1016/S0304-4076(98)00084-0).
- 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>.
- Cunningham, S. 2021. *Causal Inference: The Mixtape*. New Haven, CT: Yale University Press.
- Daniels, B. 2012. crossfold: Stata module to perform k-fold cross-validation. Statistical Software Components S457426, Department of Economics, Boston College.  
<https://ideas.repec.org/c/boc/bocode/s457426.html>.
- de Paula, A. 2020. Econometric models of network formation. *Annual Review of Economics* 12: 775–799. <https://doi.org/10.1146/annurev-economics-093019-113859>.
- Deaton, A. 2010. Instruments, randomization, and learning about development. *Journal of Economic Literature* 48: 424–455.  
<https://doi.org/10.1257/jel.48.2.424>.
- 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., and P. K. Trivedi. 2002. The structure of demand for medical care: Latent class versus two-part models. *Journal of Health Economics* 21: 601–625. [https://doi.org/10.1016/S0167-6296\(02\)00008-5](https://doi.org/10.1016/S0167-6296(02)00008-5).
- 
- \_\_\_\_\_. 2006. Maximum simulated likelihood estimation of a negative binomial regression model with multinomial endogenous treatment. *Stata Journal* 6: 246–255.  
<https://doi.org/10.1177/1536867X0600600206>.

- Doherr, T. 2018. brain: Module to provide neural network. Statistical Software Components S458566, Department of Economics, Boston College. <https://ideas.repec.org/c/boc/bocode/s458566.html>.
- 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>.
- Dong, Y., and S. Shen. 2018. Testing for rank invariance or similarity in program evaluation. *Review of Economics and Statistics* 100: 78–85. [https://doi.org/10.1162/REST\\_a\\_00686](https://doi.org/10.1162/REST_a_00686).
- 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>.
- Droste, M. 2020. pylearn. GitHub. <https://github.com/mdroste/stata-pylearn>.
- Drukker, D. M. 2002. Bootstrapping a conditional moments test for normality after tobit estimation. *Stata Journal* 2: 125–139. <https://doi.org/10.1177/1536867X0200200202>.
- \_\_\_\_\_. 2008. Treatment effects highlight use of population-averaged estimates. Unpublished manuscript.
- \_\_\_\_\_. 2016. Programming an estimation command in Stata: A map to posted entries. The Stata Blog: Not Elsewhere Classified. <https://blog.stata.com/2016/01/15/programming-an-estimation-command-in-stata-a-map-to-posted-entries/>.
- \_\_\_\_\_. 2020. Lasso and post-lasso inference. In *Sage Research Methods Foundations*, ed. P. Atkinson, S. Delamont, A. Cernat, J. W. Sakshaug, and R. A. Williams. Thousand Oaks, CA: SAGE. <http://dx.doi.org/10.4135/9781526421036936676>.
- Drukker, D. M., P. H. Egger, and I. R. Prucha. 2019. Simultaneous equations models with higher-order spatial or social network interactions. [http://econweb.umd.edu/~prucha/Papers/WP\\_DMD\\_PHE\\_IRP\\_2019.pdf](http://econweb.umd.edu/~prucha/Papers/WP_DMD_PHE_IRP_2019.pdf).

- Drukker, D. M., and R. Gates. 2006. Generating Halton sequences using Mata. *Stata Journal* 6: 214–228.  
<https://doi.org/10.1177/1536867X0600600204>.
- Drukker, D. M., I. R. Prucha, and R. Raciborski. 2013. Maximum likelihood and generalized spatial two-stage least-squares estimators for a spatial-autoregressive model with spatial-autoregressive disturbances. *Stata Journal* 13: 221–241.  
<https://doi.org/10.1177/1536867X1301300201>.
- Dube, J.-P. 1999. OLS estimation for x-sectional data with location-based dependence.  
[https://economics.uwo.ca/people/conley\\_docs/data\\_GMMWithCross\\_99/x\\_ols.ado](https://economics.uwo.ca/people/conley_docs/data_GMMWithCross_99/x_ols.ado).
- Efron, B., and T. J. Hastie. 2016. *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. New York: Cambridge University Press.
- Farrell, M. H. 2015. Robust estimation of average treatment effects with possibly more covariates than observations. *Journal of Econometrics* 189: 1–23. <https://doi.org/10.1016/j.jeconom.2015.06.017>.
- Farrell, M. H., T. Liang, and S. Misra. 2021. Deep neural networks for estimation and inference. *Econometrica* 89: 181–213.  
<https://doi.org/10.3982/ECTA16901>.
- Fernández-Val, I. 2009. Fixed effects estimation of structural parameters and marginal effects in panel probit models. *Journal of Econometrics* 150: 71–85. <https://doi.org/10.1016/j.jeconom.2009.02.007>.
- Fernández-Val, I., and M. Weidner. 2016. Individual and time effects in nonlinear panel models with large  $N, T$ . *Journal of Econometrics* 192: 291–312. <https://doi.org/10.1016/j.jeconom.2015.12.014>.
- Finkelstein, A. N., S. L. Taubman, B. J. Wright, M. Bernstein, J. H. Gruber, J. P. Newhouse, H. L. Allen, K. Baicker, and Oregon Health Study Group. 2012. The Oregon health insurance experiment: Evidence from the first year. *Quarterly Journal of Economics* 127: 1057–1106. <https://doi.org/10.1093/qje/qjs020>.

- 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>.
- Firpo, S. 2007. Efficient semiparametric estimation of quantile treatment effects. *Econometrica* 75: 259–276. <https://doi.org/10.1111/j.1468-0262.2007.00738.x>.
- Firpo, S., N. M. Fortin, and T. Lemieux. 2009. Unconditional quantile regressions. *Econometrica* 77: 953–973.  
<https://doi.org/10.3982/ECTA6822>.
- Franguridi, G., B. Gafarov, and K. Wüthrich. 2020. Conditional quantile estimators: A small sample theory. ArXiv Working Paper No. arXiv:2011.03073. <https://doi.org/10.48550/arXiv.2011.03073>.
- Frölich, M., and B. Melly. 2010. Estimation of quantile treatment effects with Stata. *Stata Journal* 10: 423–457.  
<https://doi.org/10.1177/1536867X1001000309>.
- \_\_\_\_\_. 2013. Unconditional quantile treatment effects under endogeneity. *Journal of Business and Economic Statistics* 31: 346–357. <https://doi.org/10.1080/07350015.2013.803869>.
- Galiani, S., and B. Quistorff. 2017. The synth\_runner package: Utilities to automate synthetic control estimation using synth. *Stata Journal* 17: 834–849. <https://doi.org/10.1177/1536867X1801700404>.
- Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, eds. 2013. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: CRC Press.
- Gibbons, S., and H. G. Overman. 2012. Mostly pointless spatial econometrics. *Journal of Regional Science* 52: 172–191.

<https://doi.org/10.1111/j.1467-9787.2012.00760.x>.

- Glennerster, R., and K. Takavarasha. 2013. *Running Randomized Evaluations: A Practical Guide*. Princeton, NJ: Princeton University Press.
- Glewwe, P., and P. Todd. 2022. *Impact Evaluation in International Development: Theory, Methods and Practice*. Washington, DC: World Bank Group.
- Goodman-Bacon, A. 2018. Differences-in-differences with variation in treatment timing. NBER Working Paper No. 25018, The National Bureau of Economic Research. <https://doi.org/10.3386/w25018>.
- Gould, W. W., J. Pitblado, and B. P. Poi. 2010. *Maximum Likelihood Estimation with Stata*. 4th ed. College Station, TX: Stata Press.
- Graham, B. S. 2020. Network data. In Vol. 7A of *Handbook of Econometrics*, ed. S. N. Durlauf, L. P. Hansen, J. J. Heckman, and R. L. Matzkin, 111–218. Amsterdam: Elsevier.  
<https://doi.org/10.1016/bs.hoe.2020.05.001>.
- Graham, B. S., and A. de Paula. 2020. *The Econometric Analysis of Network Data*. London: Academic Press.
- Greene, W. H. 2018. *Econometric Analysis*. 8th ed. New York: Pearson.
- Guenther, N., and M. Schonlau. 2016. Support vector machines. *Stata Journal* 16: 917–937. <https://doi.org/10.1177/1536867X1601600407>.
- Guimarães, P. 2014. poi2hdfe: Stata module to estimate a Poisson regression with two high-dimensional fixed effects. Statistical Software Components S457777, Department of Economics, Boston College. <https://ideas.repec.org/c/boc/bocode/s457777.html>.
- Guimarães, P. 2008. The fixed effects negative binomial model revisited. *Economics Letters* 99: 63–66.  
<https://doi.org/10.1016/j.econlet.2007.05.030>.
- Hagemann, A. 2017. Cluster-robust bootstrap inference in quantile regression models. *Journal of the American Statistical Association* 112: 446–456. <https://doi.org/10.1080/01621459.2016.1148610>.

- Hahn, J., and W. K. Newey. 2004. Jackknife and analytical bias reduction for nonlinear panel models. *Econometrica* 72: 1295–1319.  
<https://doi.org/10.1111/j.1468-0262.2004.00533.x>.
- Hahn, J., P. Todd, and W. Van der Klaauw. 2001. Identification and estimation of treatment effects with a regression-didcontinuity design. *Econometrica* 69: 201–209. <https://doi.org/10.1111/1468-0262.00183>.
- Hansen, B. E. 2022. *Econometrics*. Princeton: Princeton University Press.
- Härdle, W., and E. Mammen. 1993. Comparing nonparametric versus parametric regression fits. *Annals of Statistics* 21: 1926–1947.  
<https://doi.org/10.1214/aos/1176349403>.
- Hasebe, T. 2013. Copula-based maximum-likelihood estimation of sample-selection models. *Stata Journal* 13: 547–573.  
<https://doi.org/10.1177/1536867X1301300307>.
- Hastie, T. J., and R. J. Tibshirani. 1990. *Generalized Additive Models*. London: Chapman & Hall/CRC.
- 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. 2010. Building bridges between structural and program evaluation approaches to evaluating policy. *Journal of Economic Literature* 48: 356–398. <https://doi.org/10.1257/jel.48.2.356>.
- Heckman, J. J., and E. J. Vytlacil. 2005. Structural equations, treatment effects, and econometric policy evaluation. *Econometrica* 73: 699–738. <https://doi.org/10.1111/j.1468-0262.2005.00594.x>.
- Hedström, P., and T. Grund. Forthcoming. *An Introduction to Social Network Analysis and Agent-Based Modeling Using Stata*. College Station, TX: Stata Press.
- Herriges, J. A., and C. L. Kling. 1999. Nonlinear income effects in random utility models. *Review of Economics and Statistics* 81: 62–72.  
<https://doi.org/10.1162/003465399767923827>.

- Hilbe, J. M., and J. W. Hardin. 2005a. hnblogit: Stata module to estimate negative binomial-logit hurdle regression. Statistical Software Components S456401, Department of Economics, Boston College. <https://ideas.repec.org/c/boc/bocode/s456401.html>.
- \_\_\_\_\_. 2005b. hplogit: Stata module to estimate Poisson-logit hurdle regression. Statistical Software Components S456405, Department of Economics, Boston College. <https://ideas.repec.org/c/boc/bocode/s456405.html>.
- Hoechle, D. 2007. Robust standard errors for panel regressions with cross-sectional dependence. *Stata Journal* 7: 281–312. <https://doi.org/10.1177/1536867X0700700301>.
- Hoerl, A. E., and R. W. Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* 12: 55–67. <https://doi.org/10.1080/00401706.1970.10488634>.
- Hole, A. R. 2007. Fitting mixed logit models by using maximum simulated likelihood. *Stata Journal* 7: 388–401. <https://doi.org/10.1177/1536867X0700700306>.
- Holland, P. W. 1986. Statistics and causal inference. *Journal of the American Statistical Association* 81: 945–970. <https://doi.org/10.2307/2289064>.
- Honoré, B. E., and E. Kyriazidou. 2000. Panel data discrete choice models with lagged dependent variables. *Econometrica* 68: 839–874. <https://doi.org/10.1111/1468-0262.00139>.
- Hosmer, D. W., Jr., and S. Lemeshow. 1980. Goodness-of-fit tests for the multiple logistic regression model. *Communications in Statistics—Theory and Methods* 9: 1043–1069. <https://doi.org/10.1080/03610928008827941>.
- 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, M. 2019. An introduction to flexible methods for policy evaluation effects. SES Working Paper 504, University of Fribourg.
- Huber, M., and K. Wüthrich. 2019. Local average and quantile treatment effects under endogeneity: A review. *Journal of Econometrics Methods* 8: 20170007. <https://doi.org/10.1515/jem-2017-0007>.
- Ichimura, H. 1993. Semiparametric least squares (SLS) and weighted SLS estimation of single-index models. *Journal of Econometrics* 58: 71–120. [https://doi.org/10.1016/0304-4076\(93\)90114-K](https://doi.org/10.1016/0304-4076(93)90114-K).
- Imbens, G. W. 2010. Better LATE than nothing: Some comments on Deaton (2009) and Heckman and Urzua (2009). *Journal of Economic literature* 48: 399–423. <https://doi.org/10.1257/jel.48.2.399>.
- Imbens, G. W., and D. B. Rubin. 2015. *Causal Inference in Statistics, Social, and Biomedical Sciences*. Cambridge: Cambridge University Press.
- Jackson, M. O. 2010. *Social and Economic Networks*. Princeton: Princeton University Press.
- 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. 2017. kmatch: Stata module for multivariate-distance and propensity-score matching, including entropy balancing, inverse probability weighting, (coarsened) exact matching, and regression adjustment. Statistical Software Components S458346, Department of Economics, Boston College.  
<https://econpapers.repec.org/software/bocbocode/s458346.htm>.
- Jenkins, S. P. 2005. Survival Analysis. Unpublished manuscript.  
<https://www.iser.essex.ac.uk/resources/survival-analysis-with-stata>.
- Jolliffe, D., B. Krushelnitskyy, and A. Semykina. 2000. sg153: Censored least absolute deviations estimator: CLAD. *Stata Technical Bulletin* 58: 13–16. Reprinted in *Stata Technical Bulletin Reprints*. Vol. 10, pp. 240–244. College Station, TX: Stata Press.

- Kass, R. E., and A. E. Raftery. 1995. Bayes factors. *Journal of the American Statistical Association* 90: 773–795.  
<https://doi.org/10.1080/01621459.1995.10476572>.
- Kelejian, H. H., and I. R. Prucha. 2007. HAC estimation in a spatial framework. *Journal of Econometrics* 140: 131–154.  
<https://doi.org/10.1016/j.jeconom.2006.09.005>.
- Keshk, O. M. G. 2003. cdsimeq: A program to implement two-stage probit least squares. *Stata Journal* 3: 157–167.  
<https://doi.org/10.1177/1536867X0300300205>.
- Knittel, C. R., and K. Metaxoglou. 2014. Estimation of random-coefficient demand models: Two empiricists' perspective. *Review of Economics and Statistics* 96: 34–59.  
[https://doi.org/10.1162/REST\\_a\\_00394](https://doi.org/10.1162/REST_a_00394).
- Koop, G. 2003. *Bayesian Econometrics*. Chichester, UK: Wiley.
- Kowalski, A., Y. Tran, and L. Ristovska. 2016. mtebinary: Stata module to compute marginal treatment effects (MTE) with a binary instrument. Statistical Software Components S458285, Department of Economics, Boston College. <https://ideas.repec.org/c/boc/bocode/s458285.html>.
- Lancaster, T. 1990. *The Econometric Analysis of Transition Data*. Cambridge: Cambridge University Press.  
<https://doi.org/10.1017/CCOL0521265967>.
- Lechner, M. 2011. The estimation of causal effects by difference-in-difference methods. *Foundations and Trends in Econometrics* 4: 165–224. <https://doi.org/10.1561/0800000014>.
- Lee, D. S., and T. Lemieux. 2010. Regression discontinuity designs in economics. *Journal of Economic Literature* 48: 281–355.  
<https://doi.org/10.1257/jel.48.2.281>.
- Lee, L.-F. 2004. Asymptotic distributions of quasi-maximum likelihood estimators for spatial autoregressive models. *Econometrica* 72: 1899–1925. <https://doi.org/10.1111/j.1468-0262.2004.00558.x>.
- Lee, L.-F., and J. Yu. 2010. Estimation of spatial autoregressive panel data models with fixed effects. *Journal of Econometrics* 154: 165–185.

<https://doi.org/10.1016/j.jeconom.2009.08.001>.

- Lee, M. 2002. *Panel Data Econometrics: Methods-of-Moments and Limited Dependent Variables*. San Diego: Academic Press.
- Leeb, H., and B. M. Pötscher. 2005. Model selection and inference: Facts and fiction. *Econometric Theory* 21: 21–59.  
<https://doi.org/10.1017/S026646605050036>.
- LeSage, J., and R. K. Pace. 2009. *Introduction to Spatial Econometrics*. Boca Raton, FL: Chapman & Hall/CRC.
- Li, Q., and J. S. Racine. 2007. *Nonparametric Econometrics: Theory and Practice*. Princeton, NJ: Princeton University Press.
- Li, Q., and P. K. Trivedi. 2016. Adverse and advantageous selection in the Medicare supplemental market: A Bayesian analysis of prescription drug expenditure. *Health Economics* 25: 192–211.  
<https://doi.org/10.1002/hec.3133>.
- Lin, Y., and Y. Jeon. 2012. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association* 101: 578–590. <https://doi.org/10.1198/016214505000001230>.
- Lindsey, C., and S. Sheather. 2010. Variable selection in linear regression. *Stata Journal* 10: 650–669.  
<https://doi.org/10.1177/1536867X1101000407>.
- \_\_\_\_\_. 2015. Best subsets variable selection in nonnormal regression models. *Stata Journal* 15: 1046–1059.  
<https://doi.org/10.1177/1536867X1501500406>.
- List, J. A., S. Sadoff, and M. Wagner. 2011. So you want to run an experiment, now what? Some simple rules of thumb for optimal experimental design. *Experimental Economics* 14: 439–457.  
<https://doi.org/10.1007/s10683-011-9275-7>.
- 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., and J. M. C. Santos Silva. 2005. Quantiles for counts. *Journal of the American Statistical Association* 100: 1226–1237. <https://doi.org/10.1198/016214505000000330>.
- \_\_\_\_\_. 2019. Quantiles via moments. *Journal of Econometrics* 213: 145–173. <https://doi.org/10.1016/j.jeconom.2019.04.009>.
- 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. 2006. lars: Stata module to perform least angle regression. Statistical Software Components S456860, Department of Economics, Boston College. <https://ideas.repec.org/c/boc/bocode/s456860.html>.
- 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>.
- Manning, W. G., J. P. Newhouse, N. Duan, E. B. Keeler, and A. Leibowitz. 1987. Health insurance and the demand for medical care: Evidence from a randomized experiment. *American Economic Review* 77: 251–277.
- McCall, B. P. 1996. Unemployment insurance rules, joblessness, and part-time work. *Econometrica* 64: 647–682. <https://doi.org/10.2307/2171865>.
- McCrary, J. 2008. Manipulation of the running variable in the regression discontinuity design: A density test. *Journal of Econometrics* 142: 698–714. <https://doi.org/10.1016/j.jeconom.2007.05.005>.
- McCulloch, R., and P. E. Rossi. 1994. An exact likelihood analysis of the multinomial probit model. *Journal of Econometrics* 64: 207–240. [https://doi.org/10.1016/0304-4076\(94\)90064-7](https://doi.org/10.1016/0304-4076(94)90064-7).
- Melly, B., and K. Wüthrich. 2017. Local quantile treatment effects. In *Handbook of Quantile Regression*, ed. R. Koenker, V. Chernozhukov, X. He, and L. Peng, 145–164. Boca Raton, FL: Chapman & Hall/CRC. <https://doi.org/10.1201/9781315120256-10>.

- Miller, D. L. 2021. An introductory guide to event study models. Working paper.
- Miranda, A. 2006. qcount: Stata program to fit quantile regression models for count data. Statistical Software Components S456714, Department of Economics, Boston College.  
<https://ideas.repec.org/c/boc/bocode/s456714.html>.
- Molinari, F. 2020. Microeconometrics with partial identification. In Vol. 7A of *Handbook of Econometrics*, ed. S. N. Durlauf, L. P. Hansen, J. J. Heckman, and R. L. Matzkin, 355–486. Amsterdam: Elsevier.  
<https://doi.org/10.1016/bs.hoe.2020.05.002>.
- Mullahy, J. 1997. Instrumental-variable estimation of count data models: Applications to models of cigarette smoking behavior. *Review of Economics and Statistics* 79: 586–593.  
<https://doi.org/10.1162/003465397557169>.
- Mullainathan, S., and J. Spiess. 2017. Machine learning: An applied econometric approach. *Journal of Economic Perspectives* 31: 87–106.  
<https://doi.org/10.1257/jep.31.2.87>.
- Mundlak, Y. 1978. On the pooling of time series and cross section data. *Econometrica* 46: 69–85. <https://doi.org/10.2307/1913646>.
- NBER. n.d. The Oregon health insurance experiment—Data. Public use data archive. <https://www.nber.org/research/data/oregon-health-insurance-experiment-data>.
- Newey, W. K. 1985. Maximum likelihood specification testing and conditional moment tests. *Econometrica* 53: 1047–1070.  
<https://doi.org/10.2307/1911011>.
- \_\_\_\_\_. 1987. Efficient estimation of limited dependent variable models with endogenous explanatory variables. *Journal of Econometrics* 36: 231–250. [https://doi.org/10.1016/0304-4076\(87\)90001-7](https://doi.org/10.1016/0304-4076(87)90001-7).
- Newman, M. 2018. *Networks*. 2nd ed. Oxford: Oxford University Press.
- Pagan, A., and A. Ullah. 1999. *Nonparametric Econometrics*. Cambridge: Cambridge University Press.

- Pagan, A., and F. Vella. 1989. Diagnostic tests for models based on individual data: A survey. *Journal of Applied Econometrics* 4: S229–S259. <https://doi.org/10.1002/jae.3950040504>.
- Pforr, K. 2014. femlogit—Implementation of the multinomial logit model with fixed effects. *Stata Journal* 14: 847–862. <https://doi.org/10.1177/1536867X1401400409>.
- Powell, J. L. 1984. Least absolute deviations estimation for the censored regression model. *Journal of Econometrics* 25: 303–325. [https://doi.org/10.1016/0304-4076\(84\)90004-6](https://doi.org/10.1016/0304-4076(84)90004-6).
- Rabe-Hesketh, S., and A. Skrondal. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.
- Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2004. Generalized multilevel structural equation modeling. *Psychometrika* 69: 167–190. <https://doi.org/10.1007/BF02295939>.
- Rivers, D., and Q. H. Vuong. 1988. Limited information estimators and exogeneity tests for simultaneous probit models. *Journal of Econometrics* 39: 347–366. [https://doi.org/10.1016/0304-4076\(88\)90063-2](https://doi.org/10.1016/0304-4076(88)90063-2).
- Robinson, P. M. 1988. Root-n-consistent semiparametric regression. *Econometrica* 56: 931–954. <https://doi.org/10.2307/1912705>.
- Roodman, D. 2011. Fitting fully observed recursive mixed-process models with cmp. *Stata Journal* 11: 159–206. <https://doi.org/10.1177/1536867X1101100202>.
- Rosenbaum, P. R., and D. B. Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70: 41–55. <https://doi.org/10.1093/biomet/70.1.41>.
- 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.
- Rubin, D. B. 1974. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology* 66: 688–701. <https://doi.org/10.1037/h0037350>.

- \_\_\_\_\_. 1978. Bayesian inference for causal effects: The role of randomization. *Annals of Statistics* 6: 34–58.  
<https://doi.org/10.1214/aos/1176344064>.
- Ruhe, C. 2019. Bootstrap pointwise confidence intervals for covariate-adjusted survivor functions in the Cox model. *Stata Journal* 19: 185–199. <https://doi.org/10.1177/1536867X19830915>.
- Santos Silva, J. M. C., and S. Tenreyro. 2006. The log of gravity. *Review of Economics and Statistics* 88: 641–658.  
<https://doi.org/10.1162/rest.88.4.641>.
- Schonlau, M. 2005. Boosted regression (boosting): An introductory tutorial and a Stata plugin. *Stata Journal* 5: 330–354.  
<https://doi.org/10.1177/1536867X0500500304>.
- Schonlau, M., and R. Y. Zou. 2020. The random forest algorithm for statistical learning. *Stata Journal* 20: 3–29.  
<https://doi.org/10.1177/1536867X20909688>.
- Seaman, S. R., and I. R. White. 2013. Review of inverse probability weighting for dealing with missing data. *Statistical Methods in Medical Research* 22: 278–295.  
<https://doi.org/10.1177/0962280210395740>.
- Skeels, C. L., and F. Vella. 1999. A Monte Carlo investigation of the sampling behavior of conditional moment tests in tobit and probit models. *Journal of Econometrics* 92: 275–294.  
[https://doi.org/10.1016/S0304-4076\(98\)00092-X](https://doi.org/10.1016/S0304-4076(98)00092-X).
- Stock, J. H., and J. H. Wright. 2000. GMM with weak identification. *Econometrica* 68: 1055–1096. <https://doi.org/10.1111/1468-0262.00151>.
- Stukel, T. A. 1988. Generalized logistic models. *Journal of the American Statistical Association* 83: 426–431. <https://doi.org/10.2307/2288858>.
- Su, F., and P. Ding. 2021. Model-assisted analyses of cluster-randomized experiments. *Journal of the Royal Statistical Society, Series B* 83: 994–1015. <https://doi.org/10.1111/rssb.12468>.

- Tibshirani, R. J. 1999. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B* 58: 267–288.  
<https://doi.org/10.1111/j.2517-6161.1996.tb02080.x>.
- Train, K. E. 2009. *Discrete Choice Methods with Simulation*. 2nd ed. Cambridge: Cambridge University Press.
- Varian, H. R. 2014. Big data: New tricks for econometrics. *Journal of Economic Perspectives* 28: 3–27. <https://doi.org/10.1257/jep.28.2.3>.
- Verardi, V., and N. Debarsy. 2012. Robinson’s square root of N consistent semiparametric regression estimator in Stata. *Stata Journal* 12: 726–735. <https://doi.org/10.1177/1536867X1201200411>.
- Verbeek, M. 2017. *A Guide to Modern Econometrics*. 5th ed. Chichester, UK: Wiley.
- Vincent, D. W. 2015. The Berry–Levinsohn–Pakes estimator of the random-coefficients logit demand model. *Stata Journal* 15: 854–880. <https://doi.org/10.1177/1536867X1501500317>.
- Vuong, Q. H. 1989. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica* 57: 307–333.  
<https://doi.org/10.2307/1912557>.
- Wager, S., and S. Athey. 2018. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association* 113: 1228–1242.  
<https://doi.org/10.1080/01621459.2017.1319839>.
- \_\_\_\_\_. 2019. Estimating treatment effects with causal forests: An application. *Observational Studies* 5: 37–51.  
<https://doi.org/10.1353/obs.2019.0001>.
- Williams, R. 2006. Generalized ordered logit/partial proportional odds models for ordinal dependent variables. *Stata Journal* 6: 58–82.  
<https://doi.org/10.1177/1536867X0600600104>.
- Wilson, P. 2015. The misuse of the Vuong test for non-nested models to test for zero-inflation. *Economics Letters* 127: 51–53.  
<https://doi.org/10.1016/j.econlet.2014.12.029>.

- 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.  
<http://doi.org/10.2139/ssrn.3906345>.
- Wüthrich, K., and Y. Zhu. Forthcoming. Omitted variable bias of lasso-based inference methods: A finite sample analysis. *Review of Economics and Statistics* . [https://doi.org/10.1162/rest\\_a\\_01128](https://doi.org/10.1162/rest_a_01128).
- 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>.
- Yujun, L. 2009. xtbalance: Stata module to transform the dataset into balanced panel data. Statistical Software Components S457094, Department of Economics, Boston College.  
<https://ideas.repec.org/c/boc/bocode/s457094.html>.
- Zou, H. 2006. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association* 101: 1418–1429.  
<https://doi.org/10.1198/016214506000000735>.

# Author index

## A

- Abadie, A., [24.12](#), [25.5.3](#), [25.6.2](#), [25.8.1](#), [25.10](#)  
Abrevaya, J., [19.10.5](#)  
Acemoglu, D., [28.9.3](#)  
Ahrens, A., [28.8.5](#), [28.10](#)  
Allen, H. L., [24.8.1](#), [24.8.2](#), [24.8.3](#), [24.9](#)  
Ambler, G., [27.8](#)  
Amemiya, T., [17.4.4](#)  
Andresen, M. E., [25.5.5](#)  
Angrist, J. D., [24.1](#), [24.12](#), [25.5](#), [25.5.1](#), [25.5.3](#), [25.8.1](#), [25.10](#)  
Anselin, L., [26.10](#)  
Arellano, M., [22.9](#)  
Athey, S., [25.6.1](#), [28.9.4](#), [28.10](#)

## B

- Baicker, K., [24.8.1](#), [24.8.2](#), [24.8.3](#), [24.9](#)  
Baker, M., [22.7.2](#)  
Baltagi, B. H., [19.11](#), [22.9](#)  
Barker, M., [27.7](#)  
Barron, M., [28.2.7](#)  
Basu, A., [25.9](#)  
Baum, C. F., [16.9](#)  
Becker, S. O., [24.9.7](#), [24.12](#)  
Belloni, A., [28.4.3](#), [28.8.1](#), [28.9](#), [28.9.3](#)  
Bernstein, M., [24.8.1](#), [24.8.2](#), [24.8.3](#), [24.9](#)  
Berry, S. T., [18.8.4](#)  
Blundell, R., [25.3.2](#)  
Bollen, K. A., [23.8](#)  
Borah, B. J., [25.9](#)  
Borgen, N. T., [25.9.2](#)

## C

- Callaway, B., [25.6.1](#)

- Calonico, S., [25.7](#), [25.7.3](#), [25.7.5](#), [25.7.6](#)  
Cameron, A. C., [16.9](#), [17.3.3](#), [18.3.1](#), [18.7.3](#), [19.3.4](#), [19.4.5](#), [19.4.6](#),  
[19.9.1](#), [19.10](#), [20.2.2](#), [20.3.2](#), [20.3.4](#), [20.4.1](#), [20.5](#), [20.7](#), [20.7.3](#), [20.10](#),  
[21.2.1](#), [21.4.2](#), [21.7.1](#), [21.10](#), [22.4.7](#), [22.6.6](#), [22.6.9](#), [22.8](#), [22.9](#), [24.1](#),  
[24.12](#), [25.5](#), [25.10](#), [28.8.8](#), [29.3](#), [29.3.6](#), [29.4.8](#), [29.5](#), [29.9.1](#), [29.12](#),  
[30.7](#)  
Canay, I. A., [22.7.2](#)  
Card, D., [25.7.8](#)  
Carlin, J. B., [29.3](#), [29.5.2](#), [29.12](#), [30.7](#)  
Carson, R. T., [19.2.6](#)  
Cattaneo, M. D., [24.10.1](#), [24.11](#), [24.12](#), [25.7](#), [25.7.3](#), [25.7.5](#), [25.7.6](#),  
[25.10](#)  
Cerulli, G., [25.10](#)  
Chen, D., [28.9.3](#)  
Cheng, T. C., [19.11.1](#), [19.11.5](#)  
Chernozhukov, V., [25.8.3](#), [25.9.3](#), [28.4.3](#), [28.8.1](#), [28.9](#), [28.9.3](#), [28.9.4](#),  
[28.10](#)  
Chetverikov, D., [28.9.4](#), [28.10](#)  
Chib, S., [30.4.4](#)  
Cleves, M., [21.4.7](#), [21.10](#)  
Conley, T. G., [26.6.1](#), [26.10](#)  
Cruz-Gonzalez, M., [17.7](#), [22.4.8](#), [22.4.13](#), [22.4.14](#)

## D

- Daniels, B., [28.2.6](#)  
Deaton, A., [24.4.8](#)  
Deb, P., [20.5](#), [20.5.4](#), [20.10](#), [22.3](#)  
Debarsy, N., [27.6](#)  
Demirer, M., [28.9.4](#), [28.10](#)  
Diamond, A., [25.6.2](#)  
Doherr, T., [28.7.2](#)  
Donald, S. G., [19.10.5](#)  
Dong, Y., [25.8.4](#)  
Driscoll, J. C., [26.9](#)  
Drukker, D. M., [16.9](#), [18.7.3](#), [19.4.6](#), [22.4.11](#), [24.11](#), [24.12](#), [26.10](#)  
Duan, N., [22.3](#)  
Dube, J.-P., [26.6.1](#)

Duflo, E., [28.9.4](#), [28.10](#)

Dunson, D. B., [29.3](#), [29.5.2](#), [29.12](#), [30.7](#)

## E

Egger, Peter H, [26.10](#)

## F

Farrell, M. H., [25.7](#), [25.7.3](#), [25.7.5](#), [25.7.6](#), [28.9.4](#)

Fernandez-Val, I., [17.7](#), [22.4.8](#), [22.4.13](#), [22.4.14](#), [25.8.3](#), [25.9.3](#)

Finkelstein, A. N., [24.8.1](#), [24.8.2](#), [24.8.3](#), [24.9](#)

Finlay, K., [19.9.1](#)

Firpo, S., [25.9.1](#), [25.9.2](#), [25.9.3](#)

Fortin, N. M., [25.9.2](#)

Franguridi, G., [25.8.3](#)

Freese, J., [17.5.7](#), [20.3.4](#)

Friedman, J., [28.3.2](#), [28.6](#)

Frölich, M., [25.8](#), [25.8.2](#), [25.9](#), [25.9.1](#), [25.9.4](#)

## G

Gafarov, B., [25.8.3](#)

Galiani, S., [25.6.2](#)

Gates, R., [18.7.3](#)

Gelman, A., [29.3](#), [29.5.2](#), [29.12](#), [30.7](#)

Glennerster, R., [24.1](#), [24.3.6](#)

Goodman-Bacon, A., [25.6.1](#)

Gould, W., [16.3.2](#), [16.7.2](#), [16.9](#), [21.4.7](#), [21.10](#)

Greene, W. H., [16.3.2](#), [16.9](#)

Gruber, J. H., [24.8.1](#), [24.8.2](#), [24.8.3](#), [24.9](#)

Guenther, N., [28.6.7](#)

Guimaraes, P., [20.8](#)

## H

Hagemann, A., [22.7.1](#)

Hahn, J., [22.4.8](#), [25.7](#)

Hainmuller, J., [25.6.2](#)

Hansen, C. B., [25.8.3](#), [28.8.1](#), [28.8.5](#), [28.9.3](#), [28.9.4](#), [28.10](#)

Härdle, W., [27.6](#)

Hasebe, T., [19.7.2](#), [19.12](#)

Hastie, T., [27.8](#), [28.2](#), [28.3](#), [28.3.2](#), [28.6](#)  
Heckman, J. J., [25.5.5](#)  
Herriges, J. A., [18.3.1](#)  
Hilbe, J. M., [20.4.2](#)  
Hoechle, D., [26.9](#)  
Hoerl, A. E., [28.3.1](#)  
Hole, A. R., [18.8.2](#)  
Holland, A. D., [24.11](#), [24.12](#)  
Holland, P. W., [24.2](#)  
Honoré, B. E., [22.4.13](#)  
Hosmer, D. W., Jr., [17.5.2](#)  
Hsiao, C., [22.9](#)  
Huber, M., [24.12](#), [25.6.1](#), [25.10](#)

## I

Ichimua, H., [27.7](#)  
Ichino, A., [24.9.7](#), [24.12](#)  
Imbens, G. W., [24.1](#), [24.3](#), [24.4.2](#), [24.6.11](#), [24.12](#), [25.5](#), [25.5.3](#), [25.5.5](#),  
[25.6.1](#), [25.8.1](#), [25.10](#)

## J

James, G., [28.2](#), [28.3](#), [28.6](#)  
Jann, B., [24.12](#)  
Jeon, Y., [28.6.5](#)  
Johnson, S., [28.9.3](#)

## K

Kass, R. E., [29.9.1](#)  
Keeler, E. B., [22.3](#)  
Kelijian, H. H., [26.6.1](#), [26.10](#)  
Kennard, R. W., [28.3.1](#)  
Keshk, O. M. G., [17.9.2](#)  
Kling, C. L., [18.3.1](#)  
Knittel, C. R., [18.8.4](#)  
Koop, G., [29.3](#), [29.9.1](#), [29.12](#), [30.3.3](#), [30.4.3](#), [30.7](#)  
Kowalski, A. E., [25.5.5](#), [25.8.3](#)  
Kraay, A. C., [26.9](#)

Krueger, A. B., [25.5](#)  
Kyriazidou, E., [22.4.13](#)

## L

Lancaster, T., [21.10](#)  
Lechner, M., [25.6.1](#)  
Lee, D. S., [25.7](#), [25.7.8](#)  
Lee, L.-F., [26.7.2](#), [26.9](#), [26.10](#)  
Lee, M., [22.9](#)  
Leeb, H., [28.3.4](#)  
Leibowitz, A., [22.3](#)  
Lemeshow, S., [17.5.2](#)  
Lemieux, T., [25.7](#), [25.9.2](#)  
LeSage, J., [26.10](#)  
Levinson, S. J., [18.8.4](#)  
Li, Q., [24.10.2](#), [25.9.3](#), [27.9](#)  
Liang, T., [28.9.4](#)  
Lin, Y., [28.6.5](#)  
Lindsey, C., [28.2.8](#)  
List, J. A., [24.3.2](#)  
Long, J. S., [17.5.7](#), [20.3.4](#)

## M

Machado, J. A. F., [20.9](#), [20.9.2](#), [25.8.3](#)  
Magnusson, L. M., [19.9.1](#), [20.7.4](#)  
Mammen, E., [27.6](#)  
Mander, A., [28.3.2](#)  
Manjon, M., [20.3.4](#)  
Manning, W. G., [22.3](#)  
Marchenko, Y. V., [21.4.7](#), [21.10](#)  
Martinez, O., [20.3.4](#)  
McCall, P. B., [21.2.1](#)  
McCrary, J., [25.7.2](#)  
McCulloch, R., [30.4.4](#)  
Melly, B., [25.8](#), [25.8.2](#), [25.8.4](#), [25.9](#), [25.9.1](#), [25.9.3](#), [25.9.4](#)  
Metaxoglou, K., [18.8.4](#)  
Miller, D. L., [25.6.1](#)

Miranda, A., [20.9.2](#)

Misra, S., [28.9.4](#)

Mullahy, J., [20.7.3](#)

## N

Newey, W. K., [17.9.3](#), [19.4.6](#), [19.9.1](#), [22.4.8](#), [28.9.4](#), [28.10](#)

Newhouse, J. P., [22.3](#), [24.8.1](#), [24.8.2](#), [24.8.3](#), [24.9](#)

## P

Pace, R. K., [26.10](#)

Pagan, A., [19.4.6](#), [27.9](#)

Pakes, A., [18.8.4](#)

Pei, Z., [25.7.8](#)

Pforr, K., [18.10](#)

Pickles, A., [23.8](#)

Pischke, J.-S., [24.1](#), [24.12](#), [25.5](#), [25.5.1](#), [25.10](#)

Pitblado, J., [16.3.2](#), [16.7.2](#), [16.9](#)

Pötscher, B. M., [28.3.4](#)

Powell, D., [22.7.2](#)

Powell, J. L., [25.3.2](#)

Prucha, I. R., [26.6.1](#), [26.10](#)

## Q

Quistorff, B., [25.6.2](#)

## R

Rabe-Hesketh, S., [22.9](#), [23.8](#)

Raciborski, R., [26.10](#)

Racine, J. S., [25.9.3](#), [27.9](#)

Raftery, A. E., [29.9.1](#)

Rivers, D., [17.9.3](#)

Robertson, P. M., [27.6](#)

Robins, J., [28.9.4](#), [28.10](#)

Robinson, J. A., [28.9.3](#)

Roodman, D., [23.8](#)

Rosenbaum, P. R., [24.6.7](#)

Rossi, P., [30.4.4](#)

Royston, P., [27.8](#)

Rubin, D. B., [24.1](#), [24.3](#), [24.4.2](#), [24.6.7](#), [24.6.11](#), [24.12](#), [25.2](#), [25.10](#),  
[29.3](#), [29.5.2](#), [29.12](#), [30.7](#)  
Ruhe, C., [21.4.5](#)

## S

Sadoff, S., [24.3.2](#)  
Sant'Anna, P. H. C., [25.6.1](#)  
Santos Silva, J. M. C., [20.2.4](#), [20.9](#), [20.9.2](#), [25.8.3](#)  
Schaffer, M. E., [19.9.1](#), [28.8.5](#), [28.10](#)  
Schnieder, E. C., [24.8.1](#)  
Schonlau, M., [28.6.7](#), [28.7.2](#)  
Seaman, S., [19.10](#)  
Sheather, S., [28.2.8](#)  
Shen, S., [25.8.4](#)  
Skeels, C. L., [19.4.6](#)  
Skrondal, A., [22.9](#), [23.8](#)  
Smith, T., [22.7.2](#)  
Spindler, M., [28.9.3](#)  
Sribney, W., [16.3.2](#), [16.7.2](#), [16.9](#)  
Stern, H. S., [29.3](#), [29.5.2](#), [29.12](#), [30.7](#)  
Stock, J. H., [20.7.4](#)  
Stukel, T. A., [17.8.2](#)  
Sun, Y., [19.2.6](#)

## T

Takavarasha, K., [24.1](#), [24.3.6](#)  
Taubman, S. L., [24.8.1](#), [24.8.2](#), [24.8.3](#), [24.9](#)  
Tenreyro, S., [20.2.4](#)  
Tibshirani, R. J., [27.8](#), [28.2](#), [28.3](#), [28.3.2](#), [28.6](#)  
Titiunik, R., [25.7](#), [25.7.3](#), [25.7.5](#), [25.7.6](#)  
Todd, P., [25.7](#)  
Train, K., [18.7.3](#), [18.12](#)  
Trivedi, P. K., [16.9](#), [17.3.3](#), [18.3.1](#), [18.7.3](#), [19.3.4](#), [19.4.5](#), [19.4.6](#), [19.9.1](#),  
[19.10](#), [19.11.1](#), [19.11.5](#), [20.2.2](#), [20.3.2](#), [20.3.4](#), [20.4.1](#), [20.5](#), [20.7](#),  
[20.7.3](#), [20.10](#), [21.2.1](#), [21.4.2](#), [21.7.1](#), [21.10](#), [22.3](#), [22.4.7](#), [22.6.6](#), [22.6.9](#),  
[22.8](#), [22.9](#), [24.1](#), [24.10.2](#), [24.12](#), [25.5](#), [25.10](#), [28.8.8](#), [29.3](#), [29.3.6](#),  
[29.4.8](#), [29.5](#), [29.9.1](#), [29.12](#), [30.7](#)

Tudon, J., [22.4.14](#)

## U

Ullah, A., [27.9](#)

## V

van der Klaauw, W., [25.7](#)

Vehtari, A., [29.3](#), [29.5.2](#), [29.12](#), [30.7](#)

Vella, F., [19.4.6](#)

Verardi, C., [27.6](#)

Verbeek, M., [19.4.5](#)

Vincent, D. W., [18.8.4](#)

Vuong, Q. H., [17.9.3](#), [20.6.6](#)

Vytlacil, E. J., [25.5.5](#)

## W

Wager, S., [28.9.4](#), [28.10](#)

Wagner, M., [24.3.2](#)

Weber, A., [25.7.8](#)

Wei, Y., [28.9](#)

Weidner, M., [17.7](#), [22.4.8](#), [22.4.13](#), [22.4.14](#)

White, I., [19.10](#)

Williams, R., [18.9.6](#)

Wilson, P., [20.6.6](#)

Witten, D., [28.2](#), [28.3](#), [28.6](#)

Wooldridge, J. M., [16.9](#), [17.9](#), [19.9.1](#), [19.10](#), [19.11](#), [19.11.3](#), [19.11.4](#),  
[20.7](#), [22.9](#), [24.1](#), [24.6.5](#), [24.12](#), [25.3.2](#), [28.8.8](#)

Wright, B. J., [24.8.1](#), [24.8.2](#), [24.8.3](#), [24.9](#)

Wuthrich, K., [25.8.3](#), [25.8.4](#)

## Y

Yogo, M., [20.7.4](#)

Young, A., [24.3.1](#)

Yu, J., [26.9](#)

Yujun, L., [19.11.2](#)

## Z

Zaslavsky, A. M., [24.8.1](#)

Zou, H., [28.4.4](#)

Zou, R., [28.7.2](#)

# Subject index

## A

accelerated failure-time model, [21.5.5](#), [21.5.7](#)  
adjusted  $R^2$ , [28.2.3](#)  
Akaike information criterion, [20.5.10](#), [23.2.1](#), [28.2.3](#), [28.3.4](#)  
ATE, *see* [treatment effects](#)  
    definition, [24.2](#), [24.5](#)  
ATET, *see* [treatment effects](#)  
    definition, [24.2](#), [24.5](#)  
augmented IPW, *see* [treatment effects](#)  
average  
    structural function, [25.3.2](#)  
    structural mean, [25.3.2](#)  
    structural probability, [25.3.2](#)  
treatment effect, [24.2](#), *see also* [treatment effects](#)

## B

backfitting algorithm, [27.8](#)  
bagging, [28.6.4](#)  
balanced data, creating, [19.11.2](#)  
bayes prefix, [29.2.1](#)  
bayesgraph command, [29.4.4](#)  
Bayesian information criterion, [20.5.10](#), [23.2.1](#), [28.2.3](#), [28.3.4](#), [28.4.4](#),  
[28.7.2](#)  
Bayesian methods, [29](#), [29.13](#), [30](#), [30.8](#)  
    Bayes factors, [29.9.1](#), [29.9.1](#)  
    Bayes rule, [29.3.1](#)  
    compared with classical approach, [29.3.6](#)  
    credible region, [29.4.3](#)  
    data augmentation, [30.4](#), [30.4.4](#), [30.5.3](#), [30.6.3](#)  
    diagnostics, [29.4.4](#)  
    different starting values, [29.4.6](#)  
    hierarchical prior application, [29.8](#), [29.8.2](#)  
    inference, [29.3.6](#), [29.4.3](#)

linear regression, [29.2](#), [29.2.2](#), [29.5](#), [29.5.3](#), [29.6](#), [29.6](#)  
marginal effects, [29.11.3](#), [29.11.3](#)  
marginal likelihood, [29.3.1](#), [29.9.1](#)  
model selection, [29.9](#), [29.9.2](#)  
normal i.i.d. example, [29.4](#), [29.4.8](#)  
pitfalls, [29.6.4](#), [29.6.5](#)  
posterior odds ratio, [29.9.2](#), [29.9.2](#)  
prediction, [29.10](#), [29.10.3](#)  
probit application, [29.11](#), [29.11.3](#), [30.2.1](#), [30.4.3](#)  
random effects, [29.8](#), [29.8.2](#)  
**bayesmh** command, [29.3.7](#), [30.2.2](#)  
    with user-provided evaluator, [30.2.3](#)  
**bayepredict** command, [29.10.2](#)  
**bayesstats**  
    **ess** command, [29.6.3](#)  
    **grubin** command, [29.4.5](#)  
    **ic** command, [29.9.1](#)  
    **model** command, [29.9.2](#)  
    **summary** command, [29.4.3](#)  
**bayestest interval** command, [29.4.3](#)  
binary outcome models, [17.1](#), [17.12](#)  
    alternative-specific random parameters logit, [17.8.3](#)  
    Bayesian example, [29.11](#), [29.11.3](#), [30.2.1](#), [30.4.3](#)  
    Bernoulli trial, [17.2](#)  
    binomial regression, [17.10.2](#)  
    classification test, [17.5.3](#)  
    clustered data, [17.7](#), [17.7](#)  
    coefficient interpretation, [17.4.4](#)  
    complementary log–log model, [17.2.2](#)  
    confusion matrix, [17.5.3](#)  
    dynamic logit model, [22.4.13](#)  
    endogenous regressors, [17.9](#), [17.9.5](#)  
    FMM logit application, [23.3.2](#), [23.3.2](#)  
    generalized linear models, [17.4.8](#)  
    generalized logit model, [17.8.2](#)  
    goodness-of-fit measures, [17.5](#), [17.5.3](#)  
    goodness-of-fit test, [17.5.2](#)

grouped-data analysis, [17.10](#), [17.10.2](#)  
heteroskedastic probit model, [17.8.1](#)  
hypothesis tests, [17.4.5](#), [17.4.6](#)  
IV probit estimator, [17.9.4](#), [17.9.5](#)  
IV probit structural, [17.9.2](#), [17.9.3](#)  
lasso, [28.4.7](#), [28.4.7](#)  
latent-variable model, [17.2.3](#), [17.2.3](#), [17.8.1](#), [17.9.2](#), [30.2.1](#)  
linear probability model, [17.2.2](#), [17.3.4](#)  
logit model, [17.2.2](#), [17.4.2](#), [17.6.4](#)  
machine learning estimator, [28.9.2](#), [28.9.2](#)  
marginal effects, [17.6](#), [17.6.4](#)  
maximum likelihood estimation, [17.3](#)  
mixed logit, [17.8.3](#)  
model comparison, [17.4.4](#), [17.4.4](#), [17.4.7](#)  
nonlinear IV estimator, [17.9.5](#)  
nonparametric logit, [17.8.4](#)  
panel-data estimators, [22.4](#), [22.4.14](#)  
predicted probabilities, [17.5.4](#), [17.5.7](#)  
predicted versus actual frequencies, [17.5.2](#)  
predicted versus actual outcome, [17.5.3](#)  
probit model, [17.2.2](#), [17.4.4](#), [17.4.4](#), [17.4.7](#), [17.5.4](#), [17.5.4](#)  
pseudo- $R^2$ , [17.5.1](#)  
robust standard errors, [17.3.3](#), [17.7](#), [17.7](#)  
ROC curve, [17.5.5](#)  
selection model, [19.6.2](#)  
shrinkage estimation, [28.4.7](#), [28.4.7](#)  
specification analysis, [17.8.1](#)  
weak instruments, [17.9.3](#)  
binomial regression, see [binary outcome models](#)  
biprobit command, [18.11.1](#)  
BLP market demand model, [18.8.4](#)  
boost command, [28.7.2](#)  
boosting, [28.6.6](#), [28.7.2](#)  
bootstrap methods  
    for nonparametric regression, [27.4.1](#)  
    wild cluster bootstrap, [25.6.1](#)  
    wild gradient bootstrap, [22.7.1](#)

`brain` command, [28.7.2](#)

## C

`cdeco` command, [25.9.3](#)

censored data, *see* tobit models; count-data models; duration models

`cfunction` option, [25.4.1](#)

`churdle` command, [20.4.2](#)

`clad` command, [19.3.6](#)

classification, [28.6.7](#), [28.6.7](#)

`clogit` command, [18.5.9](#)

`cloglog` command, [17.3.2](#)

cluster analysis, [28.6.8](#), [28.6.8](#)

`cluster kmeans` command, [28.6.8](#)

clustered data

binary outcome models, [17.7](#), [17.7](#)

count-data models, [20.8](#), [20.8](#)

lasso, [28.4.1](#), [28.4.4](#), [28.7.2](#)

mixed nonlinear models, [23.4](#), [23.4.3](#)

multinomial outcome models, [18.10](#), [18.10](#)

random-effects Bayesian, [29.8](#), [29.8.2](#)

tobit models, [19.3.7](#)

cluster-robust inference

design-based, [24.4.7](#)

for DID, [25.6.1](#)

for panel data, [22.2.7](#)

for RCT, [24.3.5](#), [24.4.6](#)

wild cluster bootstrap, [25.6.1](#)

`cmchoiceset` command, [18.5.2](#)

`cmclogit` command, [18.5.3](#)

`cmmixlogit` command, [18.8.2](#)

`cmmprobit` command, [18.7.4](#)

`cmrologit` command, [18.5.10](#)

`cmropprobit` command, [18.7.7](#)

`cmsample` command, [18.5.2](#)

`cmset` command, [18.5.2](#)

`cmsummarize` command, [18.5.2](#)

`cmtab` command, [18.5.2](#)

coefficient interpretation

in binary outcome models, [17.4.3](#)

in multinomial outcome models, [18.4.3](#), [18.5.6](#)

in Poisson model, [20.3.2](#)

`coefpath` command, [28.4.3](#)

`collapse` command, [17.10.1](#)

complier, [25.5.1](#)

conditional independence assumption, [24.5](#)

conditional mean assumption, [24.5](#)

`contrast` command, [24.10.3](#)

control function estimator, [17.9.3](#), [20.7.2](#), [20.7.2](#), [25.4.6](#)

copula models, [19.7.1](#), [19.7.2](#)

count-data models, [20.1](#), [20.11](#)

censored Poisson, [20.3.7](#)

clustered data, [20.3.2](#), [20.8](#), [20.8](#)

comparison of four distributions, [20.5.2](#)

endogenous regressors, [20.7](#), [20.7.4](#)

equidispersion, [20.2.1](#)

excess zeros, [20.4.1](#)

finite mixture models, [20.5](#), [20.5.11](#), [23.3.5](#), [23.3.5](#)

fitted probabilities, [20.3.4](#), [20.5.4](#)

frequency distribution, [20.3.1](#)

generalized NB regression, [20.3.5](#)

GMM estimators, [20.7.3](#), [20.7.3](#)

goodness of fit, [20.3.4](#)

hurdle model, [20.4.1](#), [20.4.3](#)

interpretation of coefficients, [20.3.2](#)

lasso, [28.4.7](#), [28.4.7](#)

log-linear model alternative, [20.2.4](#)

machine learning estimator, [28.9.1](#), [28.9.1](#)

main features of count data, [20.2](#)

marginal effects, [20.3.2](#), [20.3.3](#), [20.4.3](#), [20.6.2](#)

model selection, [20.5.10](#), [20.6.6](#)

NB1 model, [20.2.2](#)

NB2 model, [20.2.2](#), [20.3.3](#), [20.3.4](#)

negative binomial distribution, [20.2.2](#)

negative binomial model, [20.3.3](#), [20.3.4](#)

nonlinear least squares, [20.3.6](#), [20.3.6](#)  
overdispersion, [20.2.2](#)  
overfitting, [20.5.11](#)  
panel-data estimators, [22.6](#), [22.6.9](#)  
point-mass distribution, [20.6.5](#), [23.3.6](#), [23.3.6](#)  
Poisson distribution, [20.2](#)  
Poisson for continuous nonnegative dependent variable, [20.2.4](#)  
Poisson mixed effects, [23.4.2](#), [23.4.3](#)  
Poisson model, [20.3.2](#), [20.3.4](#)  
quantile count regression, [20.9](#), [20.9.5](#)  
robust standard errors, [20.3.2](#), [20.3.2](#)  
shrinkage estimation, [28.4.7](#), [28.4.7](#)  
test of endogeneity, [20.7.2](#)  
test of overdispersion, [20.3.2](#)  
treatment effects, [25.4.3](#), [25.4.5](#)  
truncated Poisson and NB2, [20.3.7](#)  
two-part model, [20.4.1](#), [20.4.3](#)  
unobserved heterogeneity, [20.2.2](#)  
weak instruments, [20.7.4](#)  
zero-inflated models, [20.6](#), [20.6.6](#)  
zero-truncated models, [20.3.7](#)  
counterfactual command, [25.9.3](#)  
countfit command, [20.3.4](#)  
Cox proportional hazards model, *see* [duration models](#)  
cpoisson command, [20.3.7](#)  
cquad commands, [22.4.13](#)  
cross-fit partialing-out estimator, [28.8.9](#), [28.8.9](#)  
crossfold command, [28.2.6](#)  
cross-validation, [28.2.5](#), [28.2.8](#)  
    *K*-fold, [28.2.6](#), [28.2.6](#)  
    leave-one-out, [27.2.4](#), [28.2.7](#)  
    single split, [28.2.5](#)  
    test sample, [28.2.5](#)  
    training sample, [28.2.5](#)  
cumulative hazard function, [21.3.3](#)  
    Nelson–Aalen estimate, [21.3.3](#)  
cvplot command, [28.4.3](#)

## D

- data augmentation, [30.4](#) , [30.4.4](#) , [30.5.4](#) , [30.6.3](#)
  - for multiple imputation, [30.5.4](#) , [30.6.3](#)
- data summary examples
  - for duration data, [21.2.1](#) , [21.3.3](#)
  - for multinomial data, [18.3](#) , [18.3.3](#)
- dataset description
  - Acemoglu–Johnson–Robinson data for lasso example, [28.9.3](#)
  - American Community Survey earnings Bayesian example, [29.2.2](#)
  - cigarette sales synthetic control example, [25.6.2](#)
  - fishинг-mode multinomial data, [18.3.1](#)
  - homicide spatial data example, [26.3](#)
  - HRS private health insurance binary data, [17.4.1](#)
  - Medical Expenditure Panel Survey
    - ambulatory for tobit, [19.3.1](#)
    - doctor-visits count data, [20.3.1](#)
    - emergency room visits, [20.6.1](#)
    - for count QR example, [20.9.3](#)
    - for GSEM example, [23.6.7](#)
  - Medicare multivalued treatment-effects example, [24.10.2](#)
- Medicine in Australia:
  - Balancing Employment and Life panel attrition example, [19.11.1](#)
  - OHIE treatment-effects example, [24.8.2](#)
  - Rand HIE nonlinear panel example, [22.3](#)
  - Senate elections for regression discontinuity design example, [25.7.3](#)
  - unemployment duration example, [21.2.1](#)
- defier, [25.5.1](#)
- design-based inference, [24.4.7](#)
- deviance residual, [28.4.7](#)
- didregress command, [25.6.1](#)
- difference in differences, [25.6](#) , [25.6.1](#)
  - parallel trends assumption, [25.6.1](#)
- dimension reduction principal components, [28.5.1](#) , [28.5.1](#)
- discrete-time hazards model, [21.7](#) , [21.9](#)
- discrim command, [28.6.7](#)
- discriminant analysis, [28.6.7](#)
- double selection estimator, [28.8.10](#) , [28.9](#)

doubly robust ATE estimator, [24.6.5](#)  
doubly robust methods, *see* [treatment effects](#)  
`dslogit` command, [28.9.2](#)  
`dspoisson` command, [28.9.1](#)  
`dsregress` command, [28.8.10](#)  
duration data, *see* [duration models](#)  
duration models, [21](#), [21.11](#)

- accelerated failure-time models, [21.5.5](#), [21.5.7](#)
- censoring, [21.1](#)
- clustered data, [21.9](#)
- competing risks, [21.4.7](#), [21.4.7](#)
- complete spells, [21.1](#)
- Cox proportional hazards model, [21.4.2](#), [21.4.6](#), [21.7.1](#)
- cumulative hazard function, [21.3.3](#)
- data summary, [21.2](#), [21.2.3](#)
- diagnostics, [21.4.6](#), [21.4.6](#)
- discrete-time hazards model, [21.7](#), [21.9](#)
- exponential model, [21.1](#), [21.5.4](#)
- frailty in models, [21.5.8](#), [21.5.8](#)
- fully parametric models, [21.5](#), [21.5.8](#)
- generalized gamma model, [21.5.4](#)
- Gompertz model, [21.5.4](#)
- hazard function, [21.3.3](#)
- Kaplan–Meier estimate, [21.3.2](#)
- log-integrated hazard curve, [21.4.6](#)
- log–log survivor curve, [21.4.6](#)
- loglogistic model, [21.5.5](#)
- lognormal model, [21.5.5](#)
- marginal effects, [21.5.3](#), [21.5.8](#), [21.7.1](#)
- multiple-records data, [21.6](#), [21.8](#)
- Nelson–Aalen estimate, [21.3.3](#)
- prediction, [21.4.4](#), [21.5.3](#)
- proportional hazards model, [21.4.2](#), [21.5.4](#), [21.5.4](#), [21.5.5](#)
- Schoenfeld residual, [21.4.6](#)
- survivor function, [21.3.2](#)
- time-varying regressors, [21.8](#)
- Weibull FMM application, [23.3.7](#), [23.3.7](#)

## Weibull model, [21.5.2](#)

### E

eintreg command, [19.9.2](#), [19.12](#), [24.7.2](#), [25.3.1](#)

elastic net, [28.3.3](#), [28.3.3](#), [28.4.5](#), [28.4.5](#)

elasticnet command, [28.4.1](#), [28.4.5](#)

endogenous regressors

endogenous-switching regression model, [19.9.3](#)

ERM commands, [23.7](#), [23.7.3](#), [25.3](#), [25.3.3](#)

ET commands, [25.4](#), [25.4.6](#)

in binary outcome models, [17.9](#), [17.9.5](#)

in count-data models, [20.7](#), [20.7.4](#)

in tobit model, [19.9](#), [19.9.3](#)

nonlinear instrumental variables, [16.8.1](#)

panel data, [22.8](#), [22.8](#)

quantile treatment effects, [25.8](#), [25.8.4](#)

structural equation model, [23.6.5](#), [23.6.6](#)

treatment effects, [25](#), [25.11](#)

endogenous-switching regression model, [19.9.3](#)

eoprobit command, [23.7.1](#), [24.7.2](#), [25.3.1](#)

eprobit command, [17.9.3](#), [23.7.1](#), [24.7.2](#), [25.3.1](#)

eregress command, [19.6.2](#), [19.9.2](#), [19.12](#), [23.7.1](#), [24.7.2](#), [25.3](#)

ERM commands, [23.7](#), [23.7.1](#), [23.7.3](#)

for endogenous treatment, [25.3](#), [25.3.3](#)

for exogenous treatment, [24.7.2](#), [24.7.2](#)

for panel data, [22.8](#)

estat

alternatives command, [18.5.7](#)

classification command, [17.5.3](#)

correlation command, [18.7.5](#)

covariance command, [18.7.5](#)

gof command, [17.5.2](#)

hettest command, [19.5.3](#)

impact command, [26.7.3](#)

lcmean command, [23.3.1](#)

lcprob command, [23.3.1](#)

mfx command, [18.5.8](#)

`moran` command, [26.5](#)  
`phtest` command, [21.4.6](#)  
`teffects` command, [24.7.2](#), [25.3.1](#)  
estimation commands  
    multinomial summary, [18.2.6](#)  
    nonlinear panel summary, [22.2.6](#)  
ET commands for endogenous treatment, [25.4](#), [25.4.6](#)  
`eteffects` command, [25.4.6](#)  
`etpoisson` command, [25.4.3](#)  
`etregress` command, [25.4](#)  
event study models, [25.6.1](#)  
exponential model, [21.5.4](#)  
extended regression models, *see* [ERM commands](#)  
`extreat()` option, [24.7.2](#)

## F

failure data, *see* [duration models](#)  
`femlogit` command, [18.10](#)  
finite mixture models, [23.2](#), [23.3.8](#)  
    computational method, [23.2.2](#)  
    count-data regression, [20.5](#), [20.5.11](#)  
    definition, [23.2.1](#)  
    gamma regression, [23.3.1](#), [23.3.1](#)  
    logit regression, [23.3.2](#), [23.3.2](#)  
    marginal effects, [23.3.1](#)  
    multinomial logit, [23.3.3](#), [23.3.3](#)  
    multiple response example, [23.6.7](#), [23.6.7](#)  
    point-mass distribution, [20.6.5](#), [23.3.6](#), [23.3.6](#)  
    Poisson regression, [23.3.5](#), [23.3.5](#)  
    predicted class posterior probabilities, [23.3.1](#)  
    predicted class probabilities, [23.3.1](#)  
    predicted component means, [23.3.1](#)  
    tobit regression, [23.3.4](#), [23.3.4](#)  
    varying mixture probabilities, [23.2.1](#)  
    Weibull regression, [23.3.7](#), [23.3.7](#)  
`fmm` prefix, [20.5.3](#), [23.2.2](#)  
`foreach` command, [19.4.6](#)

`fracreg logit` command, [17.10.2](#)  
`fracreg probit` command, [17.10.2](#)  
fractional response data regression, [17.3.5](#), [17.10.2](#)

## G

`gam` command, [27.8](#)  
gamma regression model, [23.3.1](#), [23.3.1](#)  
FMM application, [23.3.1](#), [23.3.1](#)  
generalized additive model, [27.8](#), [27.8](#)  
generalized estimating equations, [22.2.3](#), [22.4.4](#)  
generalized gamma model, [21.5.4](#)  
generalized linear models  
    logit, [17.4.8](#)  
    mixed effects, [23.4.1](#), [23.4.3](#)  
    probit, [17.4.8](#)  
generalized method of moments  
    count-data example, [20.7.3](#), [20.7.4](#)  
    nonlinear example, [16.8](#), [16.8.2](#)  
generalized SEM, [23.6](#), [23.6.7](#)  
geospatial data, [26.3](#), [26.3.4](#)  
Gibbs sampling algorithm, *see* [Markov chain Monte Carlo methods](#)  
`glm` command, [17.4.8](#)  
`gmm` command, [20.7.3](#)  
`gnbreg` command, [20.3.5](#)  
`gologit2` command, [18.9.6](#)  
Gompertz model, [21.5.4](#)  
gradient methods, *see* [iterative methods](#)  
graphs  
    Bayesian diagnostics, [29.4.4](#)  
    binary outcome plot, [17.5.4](#)  
    nonparametric regression, [27.4.5](#)  
`grmap` command, [26.3.3](#)  
grouped data binary outcome models, [17.10](#), [17.10.2](#)  
`gsem` command, [23.6.2](#)  
    hurdle model, [20.4.3](#)  
`gvselect` command, [28.2.8](#)

## H

hazard function, [21.3.3](#)  
heckman command, [19.6.2](#)  
    eregress command equivalent, [19.6.2](#), [23.7.1](#)  
heckmancopula command, [19.7.2](#)  
heckoprobit command, [19.6.2](#)  
heckpoisson command, [19.6.2](#)  
heckprobit command, [19.6.2](#)  
hetoprobit command, [18.9.6](#)  
hetprobit command, [17.8.1](#)  
hierarchical models, [29.8](#), [29.8.2](#)  
hnblogit command, [20.4.3](#)  
hplogit command, [20.4.2](#)  
hurdle model, *see* [count-data models](#)  
    GSEM estimation, [20.4.3](#)  
hypothesis tests  
    permutation tests, [24.3.1](#), [25.6.2](#)

## I

identification and nonconvergence, [16.3.5](#)  
ignorability, *see* [unconfoundedness](#)  
imputation  
    inverse-probability weighting, [19.11.3](#), [19.11.3](#)  
    multiple imputation, [30.5](#), [30.6.3](#)  
    regression-based, [30.5.3](#)  
    sample selection, [19.11.4](#), [19.11.4](#)  
    summary of methods, [19.10.5](#)  
information criteria, [20.5.10](#), [23.2.1](#), [28.2.3](#), [28.3.4](#)  
instrumental variables  
    control function estimator, [17.9.3](#), [20.7.2](#), [20.7.2](#), [25.4.6](#)  
    in binary outcome models, [17.9.4](#), [17.9.5](#)  
    in count-data models, [20.7](#), [20.7.4](#)  
    in spatial regression models, [26.7.2](#), [26.8](#), [26.8](#)  
    lasso, [28.9.3](#), [28.9.3](#)  
    nonlinear IV, [16.8.1](#), [16.8.2](#), [17.9.5](#)  
    quantile regression, [25.8](#), [25.8.4](#)  
    structural model approach, [17.9.2](#), [17.9.3](#), [20.7](#), [20.7.4](#)

weak instruments, [17.9.3](#), [20.7.4](#)  
intention-to-treat effect, [24.8.1](#), [25.5.4](#)  
interval regression, *see* [tobit models](#)  
`intreg` command, [19.3.6](#)  
inverse-probability weighting, [19.10.3](#), [19.10.3](#), [19.11.3](#), [19.11.3](#), [24.6.2](#),  
[24.6.2](#)  
IPW regression adjustment, *see* [treatment effects](#)  
iterative methods, [16](#), [16.10](#)  
    checking analytical derivatives, [16.6.2](#)  
    checking data, [16.6.3](#)  
    checking parameter estimates, [16.6.5](#)  
    checking program, [16.6](#), [16.6.6](#)  
    checking standard errors, [16.6.6](#)  
    constraints on parameters, [16.3.8](#)  
    derivative methods, [16.4.1](#)  
    evaluator types, [16.4.1](#), [16.7.1](#)  
    GMM example in Mata, [16.8.2](#), [16.8.2](#)  
    gradient methods, [16.3](#), [16.3.8](#)  
    linear-form methods, [16.4.1](#)  
    Mata example, [16.2.3](#), [16.2.3](#)  
    maximization options, [16.3.1](#)  
    messages during iterations, [16.3.3](#)  
    `m1` command methods, [16.5.2](#), [16.7](#), [16.7.4](#)  
    multicollinearity, [16.6.4](#)  
    multiple optima, [16.3.6](#)  
    Newton-Raphson method, [16.2](#), [16.2.3](#)  
    NLS example, [16.5.5](#)  
    nonconvergence, [16.3.5](#)  
    nonlinear IV example in Mata, [16.8.2](#), [16.8.2](#)  
    not identified, [16.3.5](#)  
    NR example in Mata, [16.2.3](#), [16.2.3](#)  
    numerical derivatives, [16.3.7](#)  
    optimization in Mata, [16.8](#), [16.8.2](#)  
    optimization techniques, [16.3.2](#), [16.4.2](#)  
    overview of optimization tools, [16.4](#), [16.4.2](#)  
    Poisson gradient and Hessian, [16.2.2](#)  
    quadratic form methods, [16.4.1](#)

single-index model example, [16.5.3](#), [16.5.3](#)  
step-size adjustment, [16.3.2](#)  
stopping criteria, [16.3.4](#)  
two-index model example, [16.5.4](#), [16.5.4](#)  
`ivpoisson gmm` command, [20.7.3](#), [20.7.3](#)  
`ivprobit` command, [17.9.3](#)  
`ivqte` command, [25.8.2](#)  
`ivregress` command, [17.9.4](#)  
`ivtobit` command, [19.9.1](#)

## K

Kaplan–Meier estimate, [21.3.2](#), [21.3.3](#)  
kernel regression, *see* [nonparametric methods](#)  
 $K$ -fold cross-validation, [28.2.6](#), [28.2.6](#)  
`kmatch` command, [24.12](#)  
 $k$ -means clustering, [28.6.8](#)

## L

`lars` command, [28.3.2](#)  
`lasso`, [28.3.2](#), [28.3.2](#), *see also* [machine learning](#)  
adaptive lasso, [28.4.4](#), [28.4.4](#)  
application, [28.4](#), [28.4.7](#), [28.7.2](#)  
clustered data, [28.4.1](#), [28.4.4](#), [28.7.2](#)  
distribution of estimators, [28.3.4](#), [28.3.4](#)  
elastic net, [28.3.3](#), [28.3.3](#), [28.4.5](#), [28.4.5](#)  
inference, [28.8](#), [28.9.4](#), [28.10](#)  
`linear` command, [28.4.1](#)  
linear example, [28.4.2](#), [28.4.5](#)  
`logit`, [28.4.7](#), [28.4.7](#)  
`logit` command, [28.4.7](#)  
oracle property, [28.3.4](#)  
`Poisson`, [28.4.7](#), [28.4.7](#)  
`poisson` command, [28.4.7](#)  
postlasso OLS, [28.3.4](#), [28.4.3](#)  
`probit`, [28.4.7](#), [28.4.7](#)  
`probit` command, [28.4.7](#)  
`selection(bic)`, [28.4.4](#), [28.7.2](#)

sparsity assumption, [28.8.1](#)  
`lassocoef` command, [28.4.3](#)  
`lassogof` command, [28.4.3](#)  
`lassoinfo` command, [28.4.3](#)  
`lassoknots` command, [28.4.3](#)  
`lassoselect` command, [28.4.3](#)  
LATE, [25.5](#), [25.5.5](#)  
    application, [25.5.4](#), [25.5.4](#)  
    assumptions, [25.5.2](#)  
    complier, [25.5.1](#)  
    marginal treatment effects, [25.5.5](#)  
    monotonicity, [25.5.2](#)  
    Wald estimator, [25.5.1](#)  
least-absolute-deviations regression  
    censored, [19.3.6](#)  
least-angle regression, [28.3.2](#)  
leave-one-out cross-validation, [27.2.4](#), [28.2.7](#)  
linear discriminant analysis, [28.6.7](#)  
linear probability model, [17.2.2](#), [17.3.4](#)  
linear regression model  
    Bayesian example, [29.2](#), [29.2.2](#), [29.5](#), [29.5.3](#)  
    lasso example, [28.4.2](#), [28.4.5](#)  
    linear probability model, [17.3.4](#)  
local average treatment effects, *see* LATE  
local constant regression, [27.2.1](#)  
local linear regression, [27.2.2](#)  
local polynomial regression, [25.7.5](#)  
`locreg` command, [17.8.4](#)  
log-integrated hazard curve, [21.4.6](#)  
`logistic` command, [17.3.2](#)  
`logit` command, [17.3.2](#)  
logit model, *see* [binary outcome models](#)  
`logitfe` command, [22.4.8](#)  
log-linear regression  
    Poisson regression alternative, [20.2.4](#)  
log–log survivor curve, [21.4.6](#)  
loglogistic model, [21.5.5](#)

lognormal data

tobit model, [19.4](#), [19.4.7](#)

lognormal durations model, [21.5.5](#)

long-form data, [18.5.1](#), [18.5.1](#)

`loocv` command, [28.2.7](#)

`lpoly` command

compared with `npregress`, [27.4.6](#)

`lroc` command, [17.5.5](#)

## M

machine learning, [28](#), [28.11](#)

augmented IPW, [28.9.4](#)

bagging, [28.6.4](#)

big data, [28.6](#)

boosting, [28.6.6](#), [28.7.2](#)

causal analysis, [28.8](#), [28.10](#)

classification, [28.6](#), [28.6.7](#), [28.6.7](#)

cross-fit partialing out, [28.8.9](#), [28.8.9](#)

double or debiased, [28.8.9](#), [28.8.9](#)

double selection, [28.8.10](#), [28.9](#)

inference, [28](#), [28.11](#)

instrumental variables, [28.9.3](#), [28.9.3](#)

lasso, [28.3.2](#), [28.4.7](#), [28.8](#), [28.10](#)

logit, [28.4.7](#), [28.4.7](#), [28.9.2](#), [28.9.2](#)

neural networks, [28.6.2](#), [28.6.2](#), [28.7.2](#)

oracle property, [28.3.4](#)

orthogonalization, [28.8.8](#), [28.8.8](#)

overview, [28.6](#), [28.6.8](#)

partial linear model, [28.8.1](#), [28.8.10](#)

partialing out, [28.8.3](#), [28.8.8](#)

Poisson, [28.4.7](#), [28.4.7](#), [28.9.1](#), [28.9.1](#)

prediction application, [28.7](#), [28.7.3](#)

random forests, [28.6.5](#), [28.7.2](#)

regression trees, [28.6.3](#), [28.6.3](#)

shrinkage estimators, [28.3](#), [28.4.7](#)

supervised learning, [28.6](#), [28.6.1](#), [28.6.7](#)

support vector machines, [28.6.7](#)

unsupervised learning, [28.6](#), [28.6.8](#), [28.6.8](#)  
Mallow's  $C_p$  measure, [28.2.3](#)  
marginal effects  
in Bayesian model, [29.11.3](#), [29.11.3](#)  
in binary outcome models, [17.6](#), [17.6.4](#)  
in count-data models, [20.3.2](#)  
in linear mixed models, [23.4.3](#)  
in multinomial outcome models, [18.2.1](#), [18.4.5](#), [18.5.8](#), [18.8.3](#), [18.9.5](#)  
in panel logit model, [22.4.11](#), [22.4.11](#)  
in panel ordered logit model, [22.4.15](#)  
in spatial regression, [26.7.3](#), [26.7.3](#)  
in tobit model, [19.3.4](#), [19.3.4](#)  
in treatment-effects models, [25.4.1](#), [25.4.5](#)  
manual computation, [25.4.5](#)  
marginal likelihood, [29.3.1](#)  
computation, [29.9.1](#)  
marginal treatment effects, [25.5.5](#)  
margins command, [25.2.4](#), [27.4.4](#), [27.4.7](#)  
margins,  
  predict(cte) command, [25.4.1](#)  
  subpop() command, [25.4.1](#)  
marginsplot command, [27.4.5](#)  
Markov chain Monte Carlo methods, [29](#), [29.11.3](#)  
  acceptance rate, [29.4.2](#)  
  acceptance rule, [29.3.3](#), [30.3.2](#)  
  blocking parameters, [29.7.1](#), [29.7.1](#)  
  burn-in draws, [29.3.2](#)  
  convergence, [29.4.4](#), [29.6.5](#)  
  data augmentation, [30.4](#), [30.4.4](#)  
  diagnostics, [29.4.4](#), [30.3.3](#), [30.4.3](#)  
  different starting values, [29.4.5](#)  
  efficiency, [29.4.2](#), [30.3.3](#), [30.4.3](#)  
  efficiency of draws, [29.6.3](#)  
  efficiency statistic, [29.4.4](#)  
  for multiple imputation, [30.5.4](#), [30.6.3](#)  
  Gelman–Rubin statistic, [29.4.5](#)  
  Gibbs sampler example in Mata, [30.4](#), [30.4.4](#)

Gibbs sampling algorithm, [29.3.5](#), [29.3.5](#)  
Gibbs within Metropolis–Hastings, [29.7.2](#), [29.7.2](#)  
Metropolis algorithm, [29.3.3](#)  
Metropolis–Hastings algorithm, [29.3.3](#), [29.3.3](#)  
Metropolis–Hastings algorithm in Mata, [30.3](#), [30.3.3](#)  
Monte Carlo simulation error, [29.4.4](#)  
multiple chains, [29.4.5](#), [29.4.5](#)  
pitfalls, [29.6.4](#), [29.6.5](#)  
posterior draws, [29.3.2](#), [29.4.4](#), [29.4.7](#)  
prediction, [29.10](#), [29.10.3](#)  
proposal density, [29.3.3](#)  
random-walk Metropolis–Hastings algorithm, [29.3.3](#)  
sensitivity analysis, [29.4.5](#)

## Mata

Bayesian model AME, [29.11.3](#), [29.11.3](#)  
data augmentation example, [30.4](#), [30.4.4](#)  
Gibbs sampler example, [30.4](#), [30.4.4](#)  
GMM example, [16.8.2](#), [16.8.2](#)  
Metropolis–Hastings example, [30.3](#), [30.3.3](#)  
Newton-Raphson example, [16.2.3](#), [16.2.3](#)  
optimization functions, [16.8](#), [16.8.2](#)  
Poisson regression example, [16.2.3](#), [16.2.3](#)  
matching methods, *see* [treatment effects](#)  
maximization, *see* [iterative methods](#)  
maximum likelihood  
    computational methods, [16](#), [16.7.4](#)  
maximum simulated likelihood estimator, [18.7.3](#), [18.7.3](#)  
MCMC, *see* [Markov chain Monte Carlo methods](#)  
mean absolute error, [28.2.6](#)  
mean squared error, [28.2.2](#)  
measurement-error example, [23.5.7](#), [23.5.8](#)  
`meglm` command, [23.4.1](#), [23.4.2](#)  
`meintreg` command, [19.3.7](#)  
`menbreg` command, [23.4.2](#)  
`meologit` command, [18.10](#)  
`meoprobit` command, [18.10](#)  
`mepoisson` command, [23.4.2](#)

`mestreg` command, [21.9](#)  
`metobit` command, [19.3.7](#)  
Metropolis–Hastings algorithm, *see* [Markov chain Monte Carlo methods](#)

`mi`

- `convert` command, [30.6.3](#)
- `estimate` command, [30.5.5](#)
- `impute` command, [30.5.5](#)
- `misstable` command, [30.6.1](#)
- `register` command, [30.6.2](#)
- `xeq` prefix, [30.6.2](#)

minimization, *see* [iterative methods](#)

missing at random, [19.10.1](#), [30.5.1](#)  
missing completely at random, [19.10.1](#), [30.5.1](#)  
missing data, [19.10](#), [19.10.5](#)

- complete-case analysis, [19.10.2](#)
- endogenous sample selection, [19.10.4](#)
- imputation methods, [19.10.5](#), [30.5](#), [30.6.3](#)
- inverse-probability weighting, [19.10.3](#), [19.10.3](#)
- MAR, [19.10.1](#), [30.5.1](#)
- MCAR, [19.10.1](#), [30.5.1](#)
- mechanisms, [19.10.1](#), [19.10.1](#)
- MNAR, [19.10.1](#), [30.5.1](#)
- multiple imputation, [30.5](#), [30.6.3](#)
- panel attrition, [19.11](#), [19.11.5](#)
- sample selection, [19.11.4](#), [19.11.4](#)

missing not at random, [19.10.1](#), [30.5.1](#)  
missing-values imputation, [30.5](#), [30.6.3](#)  
mixed models

- generalized linear model, [23.4.1](#), [23.4.3](#)
- nonlinear, [22.4.12](#), [23.4](#), [23.4.3](#)
- Poisson example, [23.4.2](#), [23.4.3](#)

`mixlogit` command, [18.8.2](#)

`ml`

- `check` command, [16.5.3](#), [16.6.1](#)
- commands, [16.5](#), [16.7.4](#)
- `maximize` command, [16.5.1](#)
- `method d2`, [16.7.3](#)

method `gf0`, [16.7.4](#)  
method `lf`, [16.5.2](#)  
method `lf0`, [16.7](#) , [16.7.2](#)  
method `lf1`, [16.7.2](#)  
method `lf2`, [16.7.2](#)  
model command, [16.5.1](#) , [16.7.1](#)  
search command, [16.5.3](#)  
trace command, [16.6.1](#)  
`mleval` command, [16.7.1](#)  
`mlmatbysum` command, [16.7.3](#)  
`mlmatsum` command, [16.7.1](#) , [16.7.3](#)  
`mlogit` command, [18.4.1](#)  
`mlsum` command, [16.7.1](#) , [16.7.3](#)  
`mlvecsum` command, [16.7.1](#) , [16.7.3](#)  
model selection, [28.2](#) , [28.2.8](#)  
    based on predictive ability, [28.2](#) , [28.2.8](#)  
    Bayesian, [29.9](#) , [29.9.2](#)  
    best subsets, [28.2.8](#)  
    conservative model selection, [28.3.4](#)  
    consistent model selection, [28.3.4](#)  
    count-data models, [20.5.10](#)  
    cross-validation, [28.2.5](#) , [28.2.6](#)  
    information criteria, [28.2.3](#)  
    stepwise selection, [28.2.8](#)  
`mprobit` command, [18.7.2](#)  
`mtebinary` command, [25.5.5](#)  
`mtefe` command, [25.5.5](#)  
multicollinearity, [16.6.4](#)  
multinomial outcome models, [18](#) , [18.13](#)  
    additive random-utility model, [18.2.5](#) , [18.6](#)  
    alternative-specific regressors, [18.2.4](#) , [18.3.1](#) , [18.3.3](#) , [18.5](#) , [18.5.9](#) ,  
[18.7.3](#) , [18.8.3](#)  
    basic theory, [18.2.1](#) , [18.2.3](#)  
    bivariate probit model, [18.11.1](#) , [18.11.1](#)  
    BLP market demand, [18.8.4](#)  
    case-specific regressors, [18.2.4](#) , [18.3.1](#) , [18.3.2](#) , [18.5.1](#) , [18.7.2](#)  
    clustered data, [18.10](#) , [18.10](#)

`cm` commands, [18.5.2](#), [18.5.9](#), [18.7.4](#), [18.7.6](#), [18.8.2](#), [18.8.4](#)  
coefficient interpretation, [18.4.3](#), [18.5.6](#)  
conditional logit model, [18.5](#), [18.5.10](#)  
conditional versus multinomial logit, [18.5.5](#)  
data example, [18.3](#), [18.3.3](#)  
FMM application, [23.3.3](#), [23.3.3](#)  
Geweke–Hajivassiliou–Keane simulator, [18.7.3](#)  
independence of irrelevant alternatives, [18.6.1](#)  
long-form data, [18.5.1](#), [18.5.1](#)  
marginal effects, [18.2.1](#), [18.4.5](#), [18.5.8](#), [18.6.6](#), [18.7.6](#), [18.9.5](#)  
maximum likelihood estimator, [18.2.2](#)  
maximum simulated likelihood estimator, [18.7.3](#), [18.7.3](#)  
model comparison, [18.6.7](#)  
multinomial logit model, [18.4](#), [18.4.5](#)  
multinomial probit model, [18.7](#), [18.7.1](#), [18.7.7](#)  
multivariate outcomes, [18.11](#), [18.11.2](#)  
nested logit model, [18.6](#), [18.6.7](#)  
ordered logit model, [18.9.2](#)  
ordered outcome models, [18.9](#), [18.9.6](#)  
ordered probit model, [18.9.2](#)  
overview, [18.2](#), [18.2.6](#)  
predicted probabilities, [18.4.4](#), [18.5.7](#), [18.6.5](#), [18.7.6](#), [18.9.4](#), [18.11.1](#)  
prediction for new alternative, [18.5.7](#)  
probabilities, [18.2.1](#)  
random-parameters logit, [18.8](#), [18.8.4](#)  
rank-ordered logit, [18.5.10](#)  
rank-ordered probit, [18.7.7](#)  
robust standard errors, [18.2.3](#), [19.2.4](#)  
Stata commands summary, [18.2.6](#)  
wide-form data, [18.5.1](#), [18.5.1](#)  
multiple imputation, [30.5](#), [30.6.3](#)  
    inference, [30.5.2](#)  
multivalued treatment-effects example, [24.10.2](#)

## N

`nbreg` command, [20.3.3](#)  
negative binomial model, *see* [count-data models](#)

Nelson–Aalen estimate, [21.3.3](#)  
neural networks, [28.6.2](#), [28.6.2](#), [28.7.2](#)  
Newton-Raphson algorithm, *see* [iterative methods](#)  
`nlogit` command, [18.6.3](#)  
`nlogitgen` command, [18.6.3](#)  
`nlogittree` command, [18.6.3](#)  
`nlsur` command, [18.11.2](#)  
nonlinear IV, [16.8](#), [16.8.2](#)  
    additive versus multiplicative error, [20.7.3](#)  
    count-data example, [20.7.3](#), [20.7.4](#)  
nonlinear least squares  
    example, [16.5.5](#), [16.5.5](#)  
    for Poisson model, [20.3.6](#), [20.3.6](#)  
nonlinear regression  
    computational methods, [16](#), [16.10](#)  
    mixed models, [22.4.12](#), [23.4](#), [23.4.3](#)  
    nonlinear IV, [16.8](#), [16.8.2](#)  
    panel data, [22](#), [22.10](#)  
    systems of equations, [18.11.2](#)  
nonparametric methods, [27.1](#), [27.10](#)  
    bandwidth choice, [27.2.4](#), [27.2.4](#), [27.4.6](#)  
    bootstrap standard errors, [27.4.1](#)  
    cluster–robust standard errors, [27.4.1](#)  
    cross-validation, [27.2.4](#)  
    for regression discontinuity design, [25.7.2](#), [25.7.3](#)  
    graphs, [27.4.5](#)  
    kernel regression, [27.2.1](#), [27.5](#)  
    leave-one-out cross-validation, [27.2.4](#)  
    Li–Racine kernel, [27.2.3](#)  
local  
    constant, [27.2.1](#)  
    linear, [27.2.2](#)  
    linear  $m$ -estimation, [27.2.6](#)  
    logit, [17.8.4](#)  
    polynomial, [25.7.5](#)  
    multiple regressors, [27.1](#), [27.5](#), [27.5](#)  
    observations not identified, [27.4.2](#)

partial effects, [27.2.2](#), [27.4.1](#), [27.4.4](#)  
plugin bandwidth, [27.1](#)  
semiparametric regression, [27.6](#), [27.10](#)  
series regression, [27.3](#), [27.3.2](#), [27.4.7](#), [27.4.7](#)  
trimming, [27.4.3](#)  
npregress kernel command, [27.2.5](#)  
    compared with `lpoly`, [27.4.6](#)  
npregress series command, [27.3.2](#)  
numerical derivatives, [16.3.7](#)

## O

`ologit` command, [18.9.3](#)  
`oprobit` command, [18.9.6](#)  
optimization, *see* [iterative methods](#)  
`optimize()` Mata function, [16.8.2](#)  
    method `d2`, [16.8.2](#)  
ordered outcome models, *see* [multinomial outcome models](#)

## P

panel attrition, [19.11](#), [19.11.5](#)  
    inverse-probability weighting, [19.11.3](#), [19.11.3](#)  
    sample refreshment, [19.11.5](#)  
    sample selection, [19.11.4](#), [19.11.4](#)  
panel data, [22](#), [22.10](#)  
    attrition, [19.11](#), [19.11.5](#)  
    balanced panel, [19.11.2](#), [19.11.2](#)  
    balanced panel creation, [19.11.2](#)  
    between variation, [22.3.3](#)  
    bias correction, [22.4.8](#)  
    binary outcome models, [22.4](#), [22.4.14](#)  
    cluster-robust inference, [22.2.7](#)  
    correlated random-effects model, [22.2.4](#), [22.4.9](#)  
    count-data models, [22.6](#), [22.6.9](#)  
    difference in differences, [25.6.1](#)  
    dynamic  
        logit model, [22.4.13](#)  
    endogeneity, [22.3.4](#)

endogenous regressors, [22.8](#), [22.8](#)  
fixed effects, [22.2.1](#)  
fixed-effects estimator, [22.4.7](#), [22.6.6](#), [22.7.2](#)  
fixed-effects model, [22.2.1](#)  
generalized estimating equations, [22.2.3](#)  
in spatial regression models, [26.9](#), [26.9](#)  
incidental parameters problem, [22.2.1](#)  
individual-effects model, [22.2](#)  
logit model, [22.4.2](#), [22.4.15](#)  
marginal effects, [22.4.11](#), [22.4.11](#)  
missing data, [19.11](#), [19.11.5](#)  
mixed nonlinear models, [22.4.12](#)  
MNAR assumption, [19.11.4](#)  
multiplicative-effects model, [22.2](#)  
Mundlak correction, [22.2.4](#), [22.4.9](#)  
negative binomial model, [22.6.9](#), [22.6.9](#)  
nonlinear  
    example, [22.3](#), [22.3.4](#)  
    models, [22](#), [22.10](#)  
        models overview, [22.2](#), [22.2.7](#)  
    ordered logit model, [22.4.15](#), [22.4.15](#)  
    ordered probit model, [22.4.15](#), [22.4.15](#)  
    Poisson model, [22.6.1](#), [22.6.8](#)  
    pooled estimator, [22.4.2](#), [22.6.3](#), [22.7.1](#)  
    pooled model, [22.2.3](#)  
    population-averaged estimator, [22.4.5](#), [22.6.4](#)  
    population-averaged model, [22.2.3](#)  
    prediction, [22.4.11](#), [22.4.11](#), [22.6.8](#)  
    probit models, [22.4.14](#)  
    quantile regression, [22.7](#), [22.7.2](#)  
    random-coefficients model, [22.2.2](#)  
    random-effects estimator, [22.4.6](#), [22.6.5](#)  
    random-effects model, [22.2.2](#), [22.5.2](#)  
    sample refreshment, [19.11.5](#)  
    sample-selection model, [22.5.4](#), [22.5.4](#)  
    Stata nonlinear commands, [22.2.6](#)  
    tobit models, [22.5](#), [22.5.5](#)

unbalanced panel, [19.11.2](#), [19.11.2](#), [22.3.2](#)  
within variation, [22.3.3](#)  
partial identification, [19.12](#)  
partial linear model, [27.6](#), [27.6](#), [27.8](#)  
    lasso estimation, [28.8.1](#), [28.8.10](#)  
partialing-out estimator, [28.8.3](#), [28.8.8](#)  
    instrumental variables, [28.9.3](#), [28.9.3](#)  
    nonlinear models, [28.8.3](#)  
pca command, [28.5.1](#)  
poisson command, [20.3.2](#)  
Poisson model, *see* [count-data models](#)  
poivregress command, [28.9.3](#)  
population-averaged model, *see* [panel data](#)  
poregress command, [28.8.4](#)  
posterior distribution, [29.3.1](#), [29.3.1](#)  
    for normal model, [29.4.8](#), [29.5.1](#)  
    for normal regression, [29.5.3](#), [29.5.3](#)  
    MCMC draws, [29.3.2](#), [29.3.2](#)  
posterior odds ratio, [29.9.2](#), [29.9.2](#), [29.9.2](#)  
posterior predictive distribution, [29.10.1](#)  
potential outcomes, [24.2](#), [24.2](#)  
potential-outcome means, *see* [treatment effects](#)  
power calculations, [24.3.3](#), [24.3.5](#)  
    limitations, [24.3.6](#)  
power twomeans command, [24.3.3](#), [24.3.3](#)  
prchange command, [17.6.4](#)  
prcounts command, [20.3.4](#)  
precision parameter, [29.4.8](#)  
prediction, [28](#), [28.11](#)  
    Bayesian, [29.10](#), [29.10.3](#)  
    in binary outcome models, [17.5](#), [17.5.7](#)  
    in count-data models, [20.3.4](#), [20.3.4](#)  
    in duration models, [21.4.4](#), [21.5.3](#)  
    in levels from tobit in logs, [19.8](#), [19.8.3](#)  
    in nonlinear mixed models, [23.4.3](#), [23.6.6](#)  
    in panel logit model, [22.4.11](#), [22.4.11](#)  
    in panel Poisson model, [22.6.8](#)

in sample, [28.2](#), [28.2.5](#)  
in spatial regression models, [26.7.3](#), [26.7.3](#)  
in two-part model, [19.8.2](#)  
machine learning, [28](#), [28.11](#)  
machine learning application, [28.7](#), [28.7.3](#)  
of multinomial model probabilities, [18.4.4](#)  
out of sample, [28.2](#), [28.2.6](#)  
principal components, [28.5.1](#), [28.5.1](#), [28.6.8](#), [28.7.2](#)  
standardize regressors, [28.5.1](#)  
prior distribution, [29.3.1](#), [29.5.2](#), [29.5.2](#)  
can identify an unidentified model, [29.6.5](#)  
conjugate prior, [29.5.2](#)  
example of specification, [29.6.2](#)  
flat prior, [29.5.2](#), [30.2.2](#)  
hierarchical prior, [29.8](#), [29.8.2](#)  
inverse-gamma prior, [29.5.2](#)  
Jeffreys prior, [29.5.2](#)  
noninformative prior, [29.5.2](#)  
normal prior, [29.5.2](#)  
uninformative prior can be informative, [29.6.4](#)  
Wishart prior, [29.5.2](#), [30.5.4](#)  
Zellner's *g*-prior, [29.5.3](#)  
probit command, [17.3.2](#)  
probit model, *see also* [binary outcome models](#)  
probitfe command, [22.4.8](#)  
programs  
    checking program, [16.6](#), [16.6.6](#)  
propensity score, *see also* [treatment effects](#)  
    definition, [24.5](#)  
propensity-score overlap, [24.9.3](#)  
proportional hazards model, *see also* [duration models](#)  
proportions data regression, [17.3.5](#), [17.10.2](#)  
prvalue command, [17.5.7](#), [20.3.4](#)  
pscore command, [24.9.7](#)  
pseudo- $R^2$ , [17.5.1](#)  
pylearn package, [28.10](#)  
python command, [28.10](#)

## Q

`qcount` command, [20.9.2](#)  
`quadchk` command, [22.4.6](#)  
quadratic discriminant analysis, [28.6.7](#)  
quadrature methods, [22.4.6](#)  
    for multinomial probit, [18.7.2](#)  
quantile regression, [25.8](#) , [25.9.4](#)  
    conditional quantiles, [24.11](#) , [24.11](#) , [25.8.1](#) , [25.8.1](#)  
count data, [20.9](#) , [20.9.5](#)  
counterfactual analysis, [25.9.3](#)  
endogenous regressors, [25.8](#) , [25.8.4](#)  
endogenous treatment effects, [25.8](#) , [25.9.4](#)  
inverse-propensity score weighting, [25.9.1](#)  
panel data, [22.7](#) , [22.7.2](#)  
rank invariance, [25.8.4](#) , [25.8.4](#)  
rank similarity, [25.8.4](#) , [25.8.4](#)  
recentered influence function, [25.9.2](#)  
treatment effect, [24.11](#) , [24.11](#)  
unconditional quantile treatment effects, [25.9](#) , [25.9.4](#)  
unconditional quantiles, [25.9](#) , [25.9.4](#)

## R

$R^2$   
pseudo- $R^2$ , [17.5.1](#)  
random forests, [28.6.5](#) , [28.7.2](#)  
randomized controlled trials, [24.3](#) , [24.4.8](#)  
    adding covariates, [24.4](#) , [24.4.7](#)  
    assumptions, [24.3](#)  
    covariate balance, [24.3.7](#)  
    design-based, [24.4.7](#)  
    limitations, [24.4.8](#)  
    OHIE example, [24.8](#) , [24.9.8](#) , [25.5.4](#)  
    optimal sample size, [24.3.2](#)  
    power calculations, [24.3.3](#) , [24.3.5](#)  
    two-sample  $t$  test, [24.3.1](#)  
`rdbwselect` command, [25.7.6](#)  
`rdplot` command, [25.7.4](#)

`rdrobust` command, [25.7.5](#)  
receiver operator characteristics curve, [17.5.5](#)  
recentered influence function estimator, [25.9.2](#)  
recursive structure, [25.3.1](#)  
regression discontinuity design, [25.7](#), [25.7.8](#)  
    binning, [25.7.2](#)  
    fuzzy RD, [25.7.7](#), [25.7.7](#)  
    kink RD, [25.7.8](#)  
    local approach, [25.7.1](#)  
    nonparametric methods, [25.7.2](#), [25.7.3](#)  
    parametric approach, [25.7.1](#)  
    sharp RD, [25.7.1](#), [25.7.6](#)  
regression splines  
    smoothing splines, [27.8](#)  
regression trees, [28.6.3](#), [28.6.3](#)  
regressor balance, [24.9.4](#)  
`reshape` command, [18.5.1](#)  
residual augmentation approach, *see* [control function estimator](#)  
`rforest` command, [28.7.2](#)  
`rgamma()` function, [20.2.2](#)  
ridge regression, [28.3.1](#), [28.3.1](#)  
`rifreg` command, [25.9.2](#)  
    `_rmcoll` command, [16.6.4](#)  
`rnbnomial()` function, [20.2.2](#)  
root mean squared error, [28.2.6](#)  
`rpoisson()` function, [20.2.1](#), [20.2.2](#)  
`rseed()` option, [29.2.2](#)  
Rubin causal model, *see* [potential outcomes](#)

## S

sample-selection model, *see* [selection models](#)  
Schoenfeld residual, [21.4.6](#)  
seemingly unrelated regression equations  
    nonlinear, [18.11.2](#)  
selection models, [19.6](#), [19.7.2](#)  
    binary outcomes, [19.6.2](#)  
    copula models, [19.7.1](#), [19.7.2](#)

exclusion restrictions, [19.6.5](#)  
heckit model, [19.6](#)  
identification, [19.6.3](#)  
inverse of the Mills ratio, [19.6.4](#)  
maximum likelihood estimation, [19.6.2](#), [19.6.3](#)  
model definition, [19.6.1](#)  
model likelihood, [19.6.1](#)  
nonnormal selection, [19.7](#), [19.7.2](#)  
ordered outcomes, [19.6.2](#)  
panel attrition, [19.11.4](#), [19.11.4](#)  
panel sample selection, [22.5.4](#), [22.5.4](#)  
prediction, [19.8.3](#)  
sample selection, [19.10.4](#), [22.5.4](#), [22.5.4](#)  
two-step estimation, [19.6.4](#), [19.6.5](#)  
type-2 tobit, [19.6](#)  
selection on observables only, [19.10.1](#), *see also* [unconfoundedness](#)  
`sem` command, [23.5.4](#)  
SEM path builder, [23.5.6](#)  
semielasticities  
    in Poisson model, [20.3.2](#)  
`semipar` command  
    observations not identified, [27.6](#)  
semiparametric methods, [27.6](#), [27.10](#)  
    generalized additive model, [27.8](#), [27.8](#)  
    partial linear model, [27.6](#), [27.6](#), [27.8](#)  
    Robinson differencing estimator, [27.6](#)  
    semiparametric least squares, [27.7](#)  
    single-index models, [27.7](#), [27.7](#)  
    trimming, [27.6](#)  
series regression, [27.3](#), [27.3.2](#), [27.4.7](#), [27.4.7](#)  
`set`  
    seed command, [29.2.2](#)  
shrinkage estimators, [28.3](#), [28.4.7](#)  
    bias-variance tradeoff, [28.3](#)  
    elastic net, [28.3.3](#), [28.3.3](#), [28.4.5](#), [28.4.5](#)  
    lasso, [28.3.2](#), [28.4.7](#)  
    least-angle regression, [28.3.2](#)

logit, [28.4.7](#), [28.4.7](#)  
Poisson, [28.4.7](#), [28.4.7](#)  
probit, [28.4.7](#), [28.4.7](#)  
ridge regression, [28.3.1](#), [28.3.1](#)  
standardize regressors, [28.3.1](#)

simulation  
    Bayesian methods, [29](#), [29.13](#)  
    draws from negative binomial, [20.2.2](#)  
    draws from Poisson, [20.2.1](#)  
    Halton sequences, [18.7.3](#)  
    Hammersley sequences, [18.7.3](#)

simultaneous equations model  
    binary endogenous, [17.9.2](#)

single-index models  
    numerical derivatives for, [16.3.7](#)  
    semiparametric estimation, [27.7](#), [27.7](#)  
    semiparametric least squares, [27.7](#)

`skttest` command, [19.5.3](#)  
`sls` command, [27.7](#)

spatial regression, [26](#), [26.1](#), [26.11](#)  
    contiguity matrix, [26.4.1](#)  
    direct effect, [26.7.3](#)  
    FGLS with spatial errors, [26.6.2](#)  
    generalized spatial 2SLS estimation, [26.7.2](#), [26.7.4](#)  
    geospatial data, [26.3](#), [26.3.4](#)  
    geospatial distance, [26.3.4](#)  
    geospatial shapefile, [26.3.2](#)  
    heat maps, [26.3.3](#)  
    indirect effect, [26.7.3](#)  
    instrumental variables, [26.7.2](#), [26.8](#), [26.8](#)  
    inverse-distance matrix, [26.4.1](#)  
    latitude and longitude coordinates, [26.3.4](#)  
    marginal effects, [26.7.3](#), [26.7.3](#)  
    ML estimation, [26.7.2](#)  
    Moran *I* test, [26.5](#)  
    normalized weighting matrix, [26.4.1](#)  
    OLS with spatial errors, [26.6.1](#), [26.6.1](#)

overview of spatial models, [26.2](#), [26.2](#)  
panel data, [26.9](#), [26.9](#)  
planar coordinates, [26.3.4](#)  
prediction, [26.7.3](#), [26.7.3](#)  
SAR(1) in error model, [26.7.4](#), [26.7.4](#)  
SAR(1) in mean model, [26.7.2](#), [26.7.3](#)  
SARAR(1,1) model, [26.7.5](#), [26.7.5](#)  
SARAR( $p, q$ ) model, [26.7.6](#)  
spatial autoregressive models, [26.2](#), [26.2](#), [26.7](#), [26.9](#)  
spatial correlation test, [26.5](#)  
spatial dependence in error, [26.6](#), [26.6.2](#), [26.7.4](#), [26.7.4](#)  
spatial HAC standard errors, [26.6.1](#), [26.6.1](#)  
spatial lag variables, [26.4.2](#)  
spatial spillovers, [26.7.3](#)  
spatial weighting matrix, [26.2](#), [26.4](#), [26.4.2](#)  
total impact, [26.7.3](#)  
`spdistance` command, [26.3.4](#)  
specification tests  
    binary outcome models, [17.8.1](#)  
    classification test, [17.5.3](#)  
    for tobit model, [19.4.6](#), [19.4.7](#)  
    goodness-of-fit test, [17.5.2](#)  
    of endogeneity, [20.7.2](#)  
    of overdispersion, [20.3.2](#)  
    of overidentifying restrictions, [23.5.10](#)  
`spgenerate` command, [26.4.2](#)  
`spivregress` command, [26.8](#)  
`splitsample` command, [28.2.4](#)  
`spmatrix` command, [26.4](#)  
`spregress` command, [26.7.1](#)  
`spset` command, [26.3.1](#)  
`spshape2dta` command, [26.3.2](#)  
`spxtregress` command, [26.9](#)  
`sqrtlasso` command, [28.4.1](#)  
stable unit-treatment value assumption, [24.5](#)  
`st_addvar()` Mata function, [30.3.3](#)  
standardize regressors, [28.3.1](#)

for principal components, [28.5.1](#)  
`stcox` command, [21.4.1](#)  
`stcoxkm` command, [21.4.6](#)  
`stcurve` command, [21.4.5](#)  
`stdescribe` command, [21.2.3](#)  
`stjoin` command, [21.6.1](#)  
stochastic production frontier models, [22.5.5](#)  
`stofreal()` Mata function, [30.3.3](#)  
`stphplot` command, [21.4.6](#)  
`streg` command, [21.5.1](#)  
structural equation models, [23.5](#), [23.6.7](#)

- cluster-robust standard errors, [23.5.10](#)
- estimation, [23.5.1](#), [23.5.1](#), [23.5.10](#), [23.5.10](#), [23.6.1](#), [23.6.1](#)
- examples, [23.5](#), [23.5.5](#), [23.6.3](#)
- generalized SEM, [23.6](#), [23.6.7](#)
- introduction, [23.5.1](#), [23.5.1](#)
- linear IV, [23.5.9](#), [23.5.9](#)
- linear regression example, [23.5.2](#), [23.5.2](#)
- linear SEM, [23.5.1](#), [23.5.10](#)
- measurement-error example, [23.5.7](#), [23.5.8](#)
- model specification, [23.5.3](#), [23.5.3](#)
- multiple response FMM example, [23.6.7](#), [23.6.7](#)
- path diagram, [23.5.8](#)
- Poisson example, [23.6.4](#), [23.6.6](#)
- robust standard errors, [23.5.10](#)
- test of overidentifying restrictions, [23.5.10](#)
- with endogeneity, [23.6.5](#), [23.6.6](#)

structural model

- for endogenous Poisson, [20.7.1](#), [20.7.2](#)
- for endogenous probit, [17.9.2](#), [17.9.3](#)

`sts graph` command, [21.3.2](#)  
`sts list` command, [21.3.2](#)  
`stset` command, [21.2.2](#)  
`stsplit` command, [21.6.1](#)  
`st_store()` Mata function, [30.3.3](#)  
`stsum` command, [21.2.3](#)  
`stvary` command, [21.2.3](#)

supervised learning, [28.6.1](#), [28.6.7](#)  
support vector machines, [28.6.7](#)  
survival data, *see* [duration models](#)  
`svmachines` command, [28.6.7](#)  
`synth` command, [25.6.2](#)  
synthetic control, [25.6.2](#), [25.6.2](#)  
`synthrunner` command, [25.6.2](#)  
systems of equations  
nonlinear regressions, [18.11.2](#)

## T

`tebalance` command, [24.6.4](#), [24.9.4](#)  
`teffects`  
    [aipw](#) command, [24.6.5](#), [24.9.5](#), [24.10.4](#)  
    commands summary, [24.7.1](#), [24.7.1](#)  
    [ipw](#) command, [24.6.2](#), [24.9.2](#)  
    [ipwra](#) command, [24.9.5](#)  
    [nnmatch](#) command, [24.6.9](#), [24.9.8](#)  
    [psmatch](#) command, [24.6.7](#), [24.9.6](#)  
    [ra](#) command, [24.6.1](#), [24.9.1](#), [24.10.3](#)  
`telasso` command, [28.9.4](#)  
`teoverlap` command, [24.6.3](#), [24.9.3](#)  
test sample, [28.2.5](#), [28.7.1](#)  
`tnbreg` command, [20.3.7](#)  
`tobcm` command, [19.4.6](#)  
`tobit` command, [19.2.5](#)  
tobit models, [19](#), [19.13](#)  
    censored data, [19.2.1](#)  
    censored least-absolute deviations, [19.3.6](#)  
    clustered data, [19.3.7](#)  
    density, [19.2.3](#)  
    diagnostic tests, [19.4.5](#)  
    endogenous regressor, [19.9](#), [19.9.3](#)  
    endogenous-switching regression model, [19.9.3](#)  
    example, [19.3](#), [19.3.5](#)  
    FMM application, [23.3.4](#), [23.3.4](#)  
    for lognormal data, [19.4](#), [19.4.7](#)

generalized residuals, [19.4.5](#)  
interval regression, [19.3.6](#), [22.5.5](#)  
marginal effects, [19.3.4](#), [19.3.4](#)  
maximum likelihood estimation, [19.2.3](#)  
panel-data estimators, [22.5](#), [22.5.5](#)  
partial observability, [19.2.1](#)  
prediction, [19.3.3](#)  
prediction from log model, [19.8](#), [19.8.3](#)  
random-effects model, [22.5.2](#)  
selection model, [19.6](#), [19.7.2](#)  
test of heteroskedasticity, [19.4.6](#)  
test of normality, [19.4.6](#)  
tobit model, [19.2](#), [19.2.6](#)  
truncated data, [19.3.5](#)  
truncated mean, [19.2.2](#)  
two-limit tobit, [19.4.4](#)  
two-part model, [19.5](#), [19.5.3](#)  
unknown censoring point, [19.2.6](#)  
`tpoisson` command, [20.3.7](#)  
training sample, [28.2.5](#), [28.7.1](#)  
treatment effects, [24.1](#), [25.11](#)  
    assumptions, [24.5](#), [24.5](#)  
    ATE, [24.2](#), [24.5](#)  
    ATET, [24.2](#), [24.5](#)  
    augmented IPW, [24.6.5](#), [24.9.5](#), [24.10.4](#), [24.10.4](#), [28.9.4](#)  
    balancing, [24.6.4](#), [24.6.4](#)  
    blocking, [24.9.7](#)  
    complier, [25.5.1](#)  
    conditional independence assumption, [24.5](#)  
    conditional mean assumption, [24.5](#)  
    counterfactual, [24.2](#)  
    defier, [25.5.1](#)  
    difference in differences, [25.6](#), [25.6.1](#)  
    doubly robust methods, [24.6.5](#), [24.6.5](#)  
    endogenous treatment, [25](#), [25.11](#)  
    endogenous treatment parametric methods, [25.2](#), [25.4.6](#)  
    exogenous treatment, [24.1](#), [24.13](#)

heterogeneous effects, [24.6](#), [25.1](#), [25.5.1](#)  
in Poisson model, [25.4.3](#), [25.4.5](#)  
inference, [24.6.12](#), [24.6.12](#)  
intention-to-treat effect, [24.8.1](#), [25.5.4](#)  
inverse-probability weighting, [24.6.2](#), [24.6.2](#), [24.9.2](#), [24.9.4](#)  
IPW regression adjustment, [24.6.5](#), [24.9.5](#)  
lasso for AIPW, [28.9.4](#)  
LATE, [25.5](#), [25.5.5](#)  
marginal treatment effects, [25.5.5](#)  
`margins` command, [25.2.4](#), [25.4.1](#)  
matching assumption, [24.5](#)  
matching methods, [24.6.6](#), [24.6.9](#)  
monotonicity, [25.5.2](#)  
multilevel treatments, [24.10](#), [24.11](#), [25.3.3](#), [25.3.3](#)  
nearest-neighbor matching, [24.6.9](#), [24.6.9](#), [24.9.8](#), [24.9.8](#)  
OHIE example, [24.8](#), [24.9.8](#), [25.5.4](#)  
overlap assumption, [24.5](#)  
POMs, [24.6.1](#)  
`poparms` command, [24.11](#), [24.12](#)  
potential outcomes, [24.2](#), [24.2](#)  
propensity score, [24.5](#)  
propensity-score matching, [24.6.7](#), [24.6.8](#), [24.9.6](#), [24.9.7](#)  
propensity-score overlap, [24.6.3](#)  
quantile effects, [24.11](#), [24.11](#), [25.8](#), [25.9.4](#)  
quantile endogenous effects, [25.8](#), [25.8.4](#)  
regression adjustment, [24.6.1](#), [24.6.1](#), [24.9.1](#), [24.9.1](#), [24.10.3](#), [24.10.3](#)  
regression discontinuity design, [25.7](#), [25.7.8](#)  
stable unit-treatment value assumption, [24.5](#)  
synthetic control, [25.6.2](#), [25.6.2](#)  
unconfoundedness assumption, [24.5](#), [24.6.11](#), [24.6.11](#)  
triangular system, [25.3.1](#), [25.3.3](#)  
truncated data, *see* tobit models; count-data models  
`truncreg` command, [19.3.5](#)  
two-part model  
    for count data, [20.4.1](#), [20.4.3](#)  
    for log expenditure data, [19.5](#), [19.5.3](#)  
    prediction, [19.8.2](#)

two-sample  $t$  test, [24.3.1](#)

two-step estimator

sample-selection model, [19.6.4](#), [19.6.5](#)

stacked GMM standard error, [19.6.4](#)

## U

unconfoundedness, [19.10.1](#), [24.5](#), [28.8.1](#)

unobserved heterogeneity, [23](#), [23.6.7](#)

summary, [23.2](#)

unsupervised learning, [28.6.8](#), [28.6.8](#)

principal components, [28.6.8](#)

## V

variance-covariance matrix

cluster-robust, [22.2.7](#)

design-based, [24.4.7](#)

for two-step estimator, [19.6.4](#)

spatial HAC, [26.6.1](#), [26.6.1](#)

unconditional, [23.7.1](#), [25.3.1](#), [25.4.1](#)

vce (unconditional) option, [23.7.1](#), [25.3.1](#), [25.4.1](#)

vselect command, [28.2.8](#)

## W

Wald

estimator, [25.5.1](#)

Weibull model, [21.5.2](#)

FMM application, [23.3.7](#), [23.3.7](#)

wide-form data, [18.5.1](#), [18.5.1](#)

## X

x\_ols command, [26.6.1](#)

xpoivregress command, [28.9.3](#)

xporegress command, [28.8.9](#)

xtbalance command, [19.11.2](#)

xtcloglog command, [22.4](#)

xtdescribe command, [23.4](#)

xtdidregress command, [25.6.1](#)

xteintreg command, [23.7.1](#)

`xteoprobit` command, [23.7.1](#)  
`xteprobit` command, [23.7.1](#)  
`xtregress` command, [23.7.1](#)  
`xtfrontier` command, [22.5.5](#)  
`xtgee` command, [17.7](#), [22.4.4](#)  
`xtheckman` command, [22.5.4](#)  
`xtintreg` command, [22.5.5](#)  
`xtlogit` command, [17.7](#), [22.4.3](#)  
`xtlogitfe` command, [17.7](#)  
`xtnbreg` command, [22.6.1](#)  
`xtologit` command, [18.10](#), [22.4.15](#)  
`xtoprobit` command, [18.10](#)  
`xtpoisson` command, [22.6.1](#)  
`xtprobit` command, [22.4](#)  
`xtprobitfe` command, [17.7](#)  
`xtscc` command, [26.9](#)  
`xtstreg` command, [21.9](#)  
`xttobit` command, [22.5.3](#)

## Z

zero-inflated models, *see* [count-data models](#)  
zero-truncated models, *see* [count-data models](#)  
`zinb` command, [20.6.2](#)  
`zioprobit` command, [18.9.6](#)  
`zip` command, [20.6.2](#)