

Regression Models for Categorical Dependent Variables Using Stata

The background of the cover features a light gray grid. Overlaid on this grid are three parallel, downward-sloping lines. The top line is marked with circles, the middle line with diamonds, and the bottom line with squares. The lines and markers are dark gray.

Third Edition

Regression Models for Categorical Dependent Variables Using Stata

Third Edition



Regression Models for Categorical Dependent Variables Using Stata

Third Edition

J. SCOTT LONG

*Departments of Sociology and Statistics
Indiana University
Bloomington, Indiana*

JEREMY FREESE

*Department of Sociology and Institute for Policy Research
Northwestern University
Evanston, Illinois*



A Stata Press Publication
StataCorp LP
College Station, Texas



* Copyright © 2001, 2003, 2006, 2014 by StataCorp LP
All rights reserved. First edition 2001
Revised edition 2003
Second edition 2006
Third edition 2014

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

Typeset in L^AT_EX 2_ε

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN-10: 1-59718-111-0

ISBN-13: 978-1-59718-111-2

Library of Congress Control Number: 2014948009

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—without the prior written permission of StataCorp LP.

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

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

NetCourseNow is a trademark of StataCorp LP.

L^AT_EX 2_ε is a trademark of the American Mathematical Society.

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

To our parents

and their mothers

and their

and their mothers

and their mothers and their mothers and their mothers

and their mothers and their mothers

and their mothers and their mothers

and their mothers and their mothers

and their mothers and their mothers

and their mothers and their mothers and their mothers

and their mothers and their mothers

and their mothers and their mothers and their mothers

and their mothers and their mothers and their mothers

and their mothers and their mothers

and their mothers and their mothers and their mothers

and their mothers and their mothers and their mothers

and their mothers and their mothers and their mothers

and their mothers and their mothers

and their mothers and their mothers and their mothers

and their mothers and their mothers and their mothers

and their mothers and their mothers and their mothers

and their mothers and their mothers and their mothers

and their mothers and their mothers and their mothers

and their mothers and their mothers and their mothers



Contents

List of figures	xix
Preface	xxi
I General information	1
1 Introduction	7
1.1 What is this book about?	7
1.2 Which models are considered?	8
1.3 Whom is this book for?	9
1.4 How is the book organized?	9
1.5 The SPost software	11
1.5.1 Updating Stata	12
1.5.2 Installing SPost13	13
Uninstalling SPost9	14
Installing SPost13 using search	14
Installing SPost13 using net install	16
1.5.3 Uninstalling SPost13	17
1.6 Sample do-files and datasets	17
1.6.1 Installing the spost13.do package	17
1.6.2 Using spex to load data and run examples	17
1.7 Getting help with SPost	18
1.7.1 What if an SPost command does not work?	18
1.7.2 Getting help from the authors	19
What we need to help you	20
1.8 Where can I learn more about the models?	21
2 Introduction to Stata	23

2.1	The Stata interface	23
2.2	Abbreviations	27
2.3	Getting help	27
2.3.1	Online help	27
2.3.2	PDF manuals	28
2.3.3	Error messages	28
2.3.4	Asking for help	28
2.3.5	Other resources	29
2.4	The working directory	29
2.5	Stata file types	30
2.6	Saving output to log files	30
2.7	Using and saving datasets	32
2.7.1	Data in Stata format	32
2.7.2	Data in other formats	33
2.7.3	Entering data by hand	33
2.8	Size limitations on datasets	34
2.9	Do-files	34
2.9.1	Adding comments	35
2.9.2	Long lines	36
2.9.3	Stopping a do-file while it is running	37
2.9.4	Creating do-files	37
2.9.5	Recommended structure for do-files	38
2.10	Using Stata for serious data analysis	40
2.11	Syntax of Stata commands	41
2.11.1	Commands	43
2.11.2	Variable lists	43
2.11.3	if and in qualifiers	45
2.11.4	Options	46
2.12	Managing data	46
2.12.1	Looking at your data	46

2.12.2	Getting information about variables	47
2.12.3	Missing values	50
2.12.4	Selecting observations	51
2.12.5	Selecting variables	51
2.13	Creating new variables	52
2.13.1	The generate command	52
2.13.2	The replace command	54
2.13.3	The recode command	55
2.14	Labeling variables and values	56
2.14.1	Variable labels	56
2.14.2	Value labels	57
2.14.3	The notes command	59
2.15	Global and local macros	59
2.16	Loops using foreach and forvalues	61
2.17	Graphics	63
2.17.1	The graph command	65
2.18	A brief tutorial	73
2.19	A do-file template	79
2.20	Conclusion	81
3	Estimation, testing, and fit	83
3.1	Estimation	84
3.1.1	Stata's output for ML estimation	84
3.1.2	ML and sample size	85
3.1.3	Problems in obtaining ML estimates	85
3.1.4	Syntax of estimation commands	86
3.1.5	Variable lists	87
	Using factor-variable notation in the variable list	87
	Specifying interaction and polynomials	89
	More on factor-variable notation	90
3.1.6	Specifying the estimation sample	93

	Missing data	93
	Information about missing values	95
	Postestimation commands and the estimation sample	98
3.1.7	Weights and survey data	99
	Complex survey designs	100
3.1.8	Options for regression models	102
3.1.9	Robust standard errors	103
3.1.10	Reading the estimation output	105
3.1.11	Storing estimation results	107
	(Advanced) Saving estimates to a file	108
3.1.12	Reformatting output with estimates table	111
3.2	Testing	114
3.2.1	One-tailed and two-tailed tests	115
3.2.2	Wald and likelihood-ratio tests	115
3.2.3	Wald tests with test and testparm	116
3.2.4	LR tests with lrtest	118
	Avoiding invalid LR tests	120
3.3	Measures of fit	120
3.3.1	Syntax of fitstat	120
3.3.2	Methods and formulas used by fitstat	123
3.3.3	Example of fitstat	129
3.4	estat postestimation commands	130
3.5	Conclusion	131
4	Methods of interpretation	133
4.1	Comparing linear and nonlinear models	133
4.2	Approaches to interpretation	136
4.2.1	Method of interpretation based on predictions	137
4.2.2	Method of interpretation using parameters	138
4.2.3	Stata and SPost commands for interpretation	138
4.3	Predictions for each observation	138

4.4	Predictions at specified values	139
4.4.1	Why use the <code>m*</code> commands instead of <code>margins</code> ?	140
4.4.2	Using <code>margins</code> for predictions	141
	Predictions using interaction and polynomial terms	146
	Making multiple predictions	146
	Predictions for groups defined by levels of categorical variables	150
4.4.3	(Advanced) Nondefault predictions using <code>margins</code>	153
	The <code>predict()</code> option	153
	The <code>expression()</code> option	154
4.4.4	Tables of predictions using <code>mtable</code>	155
	<code>mtable</code> with categorical and count outcomes	158
	(Advanced) Combining and formatting tables using <code>mtable</code>	160
4.5	Marginal effects: Changes in predictions	162
4.5.1	Marginal effects using <code>margins</code>	163
4.5.2	Marginal effects using <code>mtable</code>	164
4.5.3	Posting predictions and using <code>mlincom</code>	165
4.5.4	Marginal effects using <code>mchange</code>	166
4.6	Plotting predictions	171
4.6.1	Plotting predictions with <code>marginsplot</code>	171
4.6.2	Plotting predictions using <code>mgen</code>	173
4.7	Interpretation of parameters	178
4.7.1	The <code>listcoef</code> command	179
4.7.2	Standardized coefficients	180
4.7.3	Factor and percentage change coefficients	184
4.8	Next steps	184
II	Models for specific kinds of outcomes	185
5	Models for binary outcomes: Estimation, testing, and fit	187
5.1	The statistical model	187
5.1.1	A latent-variable model	188

5.1.2	A nonlinear probability model	192
5.2	Estimation using logit and probit commands	192
5.2.1	Example of logit model	194
5.2.2	Comparing logit and probit	196
5.2.3	(Advanced) Observations predicted perfectly	197
5.3	Hypothesis testing	200
5.3.1	Testing individual coefficients	200
5.3.2	Testing multiple coefficients	203
5.3.3	Comparing LR and Wald tests	205
5.4	Predicted probabilities, residuals, and influential observations	206
5.4.1	Predicted probabilities using predict	206
5.4.2	Residuals and influential observations using predict	209
5.4.3	Least likely observations	216
5.5	Measures of fit	218
5.5.1	Information criteria	219
5.5.2	Pseudo- R^2 's	221
5.5.3	(Advanced) Hosmer–Lemeshow statistic	223
5.6	Other commands for binary outcomes	225
5.7	Conclusion	225
6	Models for binary outcomes: Interpretation	227
6.1	Interpretation using regression coefficients	228
6.1.1	Interpretation using odds ratios	228
6.1.2	(Advanced) Interpretation using y^*	235
6.2	Marginal effects: Changes in probabilities	239
6.2.1	Linked variables	241
6.2.2	Summary measures of change	242
	MEMs and MERs	243
	AMEs	243
	Standard errors of marginal effects	244
6.2.3	Should you use the AME, the MEM, or the MER?	244

6.2.4	Examples of marginal effects	246
	AMEs for continuous variables	248
	AMEs for factor variables	251
	Summary table of AMEs	252
	Marginal effects for subgroups	254
	MEMs and MERs	255
	Marginal effects with powers and interactions	259
6.2.5	The distribution of marginal effects	261
6.2.6	(Advanced) Algorithm for computing the distribution of effects	265
6.3	Ideal types	270
6.3.1	Using local means with ideal types	273
6.3.2	Comparing ideal types with statistical tests	274
6.3.3	(Advanced) Using macros to test differences between ideal types	275
6.3.4	Marginal effects for ideal types	278
6.4	Tables of predicted probabilities	280
6.5	Second differences comparing marginal effects	285
6.6	Graphing predicted probabilities	286
6.6.1	Using marginsplot	287
6.6.2	Using mgen with the graph command	290
6.6.3	Graphing multiple predictions	293
6.6.4	Overlapping confidence intervals	297
6.6.5	Adding power terms and plotting predictions	301
6.6.6	(Advanced) Graphs with local means	303
6.7	Conclusion	308
7	Models for ordinal outcomes	309
7.1	The statistical model	310
7.1.1	A latent-variable model	310
7.1.2	A nonlinear probability model	314
7.2	Estimation using ologit and oprobit	314

7.2.1	Example of ordinal logit model	315
7.2.2	Predicting perfectly	319
7.3	Hypothesis testing	320
7.3.1	Testing individual coefficients	321
7.3.2	Testing multiple coefficients	322
7.4	Measures of fit using fitstat	324
7.5	(Advanced) Converting to a different parameterization	325
7.6	The parallel regression assumption	326
7.6.1	Testing the parallel regression assumption using oparallel	329
7.6.2	Testing the parallel regression assumption using brant	330
7.6.3	Caveat regarding the parallel regression assumption	331
7.7	Overview of interpretation	331
7.8	Interpreting transformed coefficients	332
7.8.1	Marginal change in y^*	332
7.8.2	Odds ratios	335
7.9	Interpretations based on predicted probabilities	338
7.10	Predicted probabilities with predict	339
7.11	Marginal effects	341
7.11.1	Plotting marginal effects	344
7.11.2	Marginal effects for a quick overview	350
7.12	Predicted probabilities for ideal types	351
7.12.1	(Advanced) Testing differences between ideal types	354
7.13	Tables of predicted probabilities	355
7.14	Plotting predicted probabilities	359
7.15	Probability plots and marginal effects	364
7.16	Less common models for ordinal outcomes	370
7.16.1	The stereotype logistic model	370
7.16.2	The generalized ordered logit model	371
7.16.3	(Advanced) Predictions without using factor-variable notation	374

7.16.4	The sequential logit model	378
7.17	Conclusion	382
8	Models for nominal outcomes	385
8.1	The multinomial logit model	386
8.1.1	Formal statement of the model	390
8.2	Estimation using the mlogit command	390
	Weights and complex samples	391
	Options	391
8.2.1	Example of MNLM	392
8.2.2	Selecting different base outcomes	395
8.2.3	Predicting perfectly	397
8.3	Hypothesis testing	398
8.3.1	mlogtest for tests of the MNLM	398
8.3.2	Testing the effects of the independent variables	399
8.3.3	Tests for combining alternatives	403
8.4	Independence of irrelevant alternatives	407
8.4.1	Hausman–McFadden test of IIA	408
8.4.2	Small–Hsiao test of IIA	409
8.5	Measures of fit	411
8.6	Overview of interpretation	411
8.7	Predicted probabilities with predict	412
8.8	Marginal effects	415
8.8.1	(Advanced) The distribution of marginal effects	420
8.9	Tables of predicted probabilities	423
8.9.1	(Advanced) Testing second differences	425
8.9.2	(Advanced) Predictions using local means and subsamples	428
8.10	Graphing predicted probabilities	432
8.11	Odds ratios	435
8.11.1	Listing odds ratios with listcoef	435
8.11.2	Plotting odds ratios	436

8.12	(Advanced) Additional models for nominal outcomes	444
8.12.1	Stereotype logistic regression	445
8.12.2	Conditional logit model	454
8.12.3	Multinomial probit model with IIA	465
8.12.4	Alternative-specific multinomial probit	469
8.12.5	Rank-ordered logit model	475
8.13	Conclusion	479
9	Models for count outcomes	481
9.1	The Poisson distribution	481
9.1.1	Fitting the Poisson distribution with the poisson command	483
9.1.2	Comparing observed and predicted counts with mgen	484
9.2	The Poisson regression model	487
9.2.1	Estimation using poisson	488
	Example of the PRM	489
9.2.2	Factor and percentage changes in $E(y x)$	490
	Example of factor and percentage change	492
9.2.3	Marginal effects on $E(y x)$	493
	Examples of marginal effects	495
9.2.4	Interpretation using predicted probabilities	496
	Predicted probabilities using mtable and mchange	496
	Treating a count independent variable as a factor variable	498
	Predicted probabilities using mgen	500
9.2.5	Comparing observed and predicted counts to evaluate model specification	501
9.2.6	(Advanced) Exposure time	504
9.3	The negative binomial regression model	507
9.3.1	Estimation using nbreg	509
	NB1 and NB2 variance functions	509
9.3.2	Example of NBRM	510
9.3.3	Testing for overdispersion	511

9.3.4	Comparing the PRM and NBRM using estimates table	511
9.3.5	Robust standard errors	512
9.3.6	Interpretation using $E(y x)$	514
9.3.7	Interpretation using predicted probabilities	516
9.4	Models for truncated counts	518
9.4.1	Estimation using <code>tpoisson</code> and <code>tnbreg</code>	521
	Example of zero-truncated model	521
9.4.2	Interpretation using $E(y x)$	523
9.4.3	Predictions in the estimation sample	524
9.4.4	Interpretation using predicted rates and probabilities	525
9.5	(Advanced) The hurdle regression model	527
9.5.1	Fitting the hurdle model	528
9.5.2	Predictions in the sample	531
9.5.3	Predictions at user-specified values	533
9.5.4	Warning regarding sample specification	534
9.6	Zero-inflated count models	535
9.6.1	Estimation using <code>zinb</code> and <code>zip</code>	538
9.6.2	Example of zero-inflated models	539
9.6.3	Interpretation of coefficients	540
9.6.4	Interpretation of predicted probabilities	541
	Predicted probabilities with <code>mtable</code>	542
	Plotting predicted probabilities with <code>mgen</code>	543
9.7	Comparisons among count models	544
9.7.1	Comparing mean probabilities	545
9.7.2	Tests to compare count models	547
9.7.3	Using <code>countfit</code> to compare count models	551
9.8	Conclusion	558
	References	561
	Author index	569
	Subject index	573

Figures

2.1	The Stata user interface	24
4.1	A simple linear model	134
4.2	A simple nonlinear model	136
5.1	Relationship between latent variable y^* and $\Pr(y = 1)$ for the BRM .	189
5.2	Relationship between the linear model $y^* = \alpha + \beta x + \varepsilon$ and the nonlinear probability model $\Pr(y = 1 x) = F(\alpha + \beta x)$	191
5.3	The distinction between an outlier and an influential observation . .	210
6.1	Marginal change and discrete change in the BRM	239
6.2	Overlapping confidence intervals compared with discrete change . .	300
7.1	Relationship between observed y and latent y^* in ORM with one independent variable	311
7.2	Plot of predicted probabilities and cumulative probabilities for the ordered logit model	362
9.1	The Poisson probability density function (PDF) for different rates . .	482

2000

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

Preface

As with previous editions, our goal in writing this book is to make it routine to carry out the complex calculations necessary to fully interpret regression models for categorical outcomes. Interpreting these models is complex because the models are nonlinear. Software packages that fit these models often do not provide options that make it simple to compute the quantities that are useful for interpretation; when they do provide these options, there is usually little guidance as to how to use them. In this book, we briefly describe the statistical issues involved in interpretation and then show how you can use Stata to make these computations.

While our purpose remains the same, this third edition is an almost complete rewrite of the second edition—almost every line of code in our `SPost` commands has been rewritten. Advances in computing and the addition of new features to Stata has expanded the possibilities for routinely applying more sophisticated methods of interpretation. As a result, ideas we noted in previous editions as good in principle are now much more straightforward to implement in practice. For example, while you could compute average marginal effects using commands discussed in previous editions, it was difficult and few people did so (ourselves included). Likewise, in previous editions, we relegated methods for dealing with nonlinearities and interactions on the right-hand side of the model to the last chapter, and our impression was that few readers took advantage of these ideas because they were comparatively difficult and error-prone to use.¹

These limitations changed with the addition of factor variables and the `margins` command in Stata 11. It took us personally quite a while to fully absorb the potential of these powerful enhancements and decide how best to take advantage of them. Plus, Stata 13 added several features that were essential for what we wanted to do.

This third edition considers the same models as the second edition of the book. We still find these to be the most valuable models for categorical outcomes. And, as in previous editions, our discussion is limited to models for cross-sectional data. While we would like to consider models for panel data and other hierarchical data structures, doing so would at least double the size of an already long book.

1. Those who have read previous editions will note that this last chapter has been dropped entirely. In addition to covering linked variables of the right-hand side, that chapter also discussed adapting our commands to other estimation commands; however, this is now obsolete because `margins` works with most estimation commands. We also dropped the section on working effectively in Stata because Long's (2009) *Workflow of Data Analysis Using Stata* covers these topics in detail.

We note, however, that many of our `SPost` commands—such as `mtable`, `mgen`, and `mchange` (hereafter referred to as the `m*` commands)—are based on `margins` and can be used with any model that is supported by `margins`. This is a substantial change from our earlier `prchange`, `prgen`, `prtab`, and `prvalue` commands, which only worked with the models discussed in the book. A second major improvement is that our `m*` commands work with weights and survey estimation, because these are supported by `margins`.

`SPost` was originally developed using Stata 4 and Stata 5. Since then, our commands have often been enhanced to use new features in Stata. Sometimes these enhancements have led to code that was not as efficient, robust, or elegant as we would have liked. In `SPost13`, we rewrote much of the code, incorporated better returns, improved output, and removed obscure or obsolete features.

How to cite

Our commands are not officially part of Stata. We have written them in an effort to contribute to the community of researchers whose work involves extensive use of the models we cover. If you use our commands or other materials in published work, we ask that you cite our work in the same way that you cite other useful sources. We ask that you simply cite the book rather than providing different citations to different commands:

Long, J. S., and J. Freese. 2014. *Regression Models for Categorical Dependent Variables in Stata*. 3rd ed. College Station, TX: Stata Press.

Thanks

Hundreds of people have contributed to our work since 2001. We cannot possibly mention them all here, but we gratefully acknowledge them for taking the time to give us their ideas. Thousands of students have taken classes using previous editions of the books, and many have given us suggestions to make the book or the commands more effective.

In writing this third edition, several people deserve to be mentioned. Ian Anson and Trent Mize tested commands and provided comments on draft chapters. Tom VanHeuvelen ran labs in two classes that used early versions of the commands, helped students work around bugs, made valuable suggestions on how to improve the commands, provided detailed comments on each chapter, was a sounding board for ideas, and was an exceptional research assistant. Rich Williams gave us many suggestions that improved the book and our commands. He has a (sometimes) valuable gift for finding bugs as well. Scott Long gratefully acknowledges the support provided by the College of Arts and Sciences at Indiana University.

People at StataCorp provided their expertise in many ways. More than this, though, we are grateful for their engagement and support of our project. Jeff Pitblado was enor-

mously helpful as we incorporated factor variables and `margins` into our commands. His advice made our code far more compact and reliable. Vince Wiggins provided valuable advice on our graphing commands and helped us understand `margins` better. Lisa Gilmore, as always, did a great job moving the book from draft to print. Most importantly, discussions with David Drukker stimulated our thinking about a new edition and, as always, asked challenging questions that made our ideas better.

Illinois and Indiana
August 2014

Jeremy Freese
Scott Long

Part I

General information

Our book is about using Stata to fit and interpret regression models with categorical outcomes, with an emphasis on interpretation. The book is divided into two parts. Part I contains general information that applies to all the regression models that are then considered in detail in part II.

- **Chapter 1** is a brief orienting discussion that also includes critical information about installing a collection of Stata commands, known collectively as `SPost13`, that we have written to facilitate the interpretation of regression models. Without these commands, you cannot do many of the things we suggest in later chapters.
- **Chapter 2** includes an introduction to Stata for those who have not used the program or are just beginning to use it. In addition to this basic information on the Stata interface, the chapter includes an introduction to using macros, returns, and loops. Because these tools are used extensively in later chapters, we encourage more advanced Stata users to at least skim these sections.
- **Chapter 3** considers issues of estimation, testing, and assessing fit that are common to all the models considered in later chapters. We discuss both the statistical issues involved and the Stata commands that carry out these operations. Readers already using Stata may be familiar with much of this material. Still, we think the chapter is worth at least a quick read through, paying particular attention to factor-variable notation, which is used extensively in later chapters.
- **Chapter 4** considers commands and approaches to interpretation that are used with all the regression models in part II of the book. Most importantly, we discuss the `margins` command along with the `mtable`, `mgen`, and `mchange` commands we have written that use `margins`. You do not need to master this material on first reading, but you should at least skim each section so that you can return to relevant sections as you read later chapters.

Part II encompasses chapters 5–9 and is organized by the type of outcome being modeled. These chapters apply the methods introduced in chapters 3 and 4 using the package of commands we show you how to install in chapter 1.

We must add some words of caution: First, we have not been able to test our commands with every `margins`-compatible estimation command. We have been encouraged that our commands have worked with other models we have used in our research. Nonetheless, if you use our `m*` commands with other models, you should include the `details` option so that you can compare the output from `margins` with the summaries provided by our commands.

Second, `margins` is an extremely powerful command that has features for applications that we have not considered, such as experimental design. Our philosophy in designing our commands is to allow them to allow these options, which are passed along to `margins` to do the computation. Everything should work, but we have not been able to test every option. While we could have designed our commands to intercept all `margins` options that we have not tested, this seemed less useful than allowing you to try them. If you use options that are not discussed in the book, please let us know if you encounter problems.

Third, just because `margins` can compute a particular statistic does not mean that it is reasonable to interpret that statistic. It can estimate statistics that are valuable and appropriate for a given model, and it will also compute the same statistics for another model for which those statistics are inappropriate. The burden of what to compute is left with the user. This is more broadly true of `margins`: its great power and flexibility also put an extra burden of responsibility on the user for making sure results are interpreted correctly.

Another cost of the remarkable generality of `margins` is that very general routines are used to make computations. These routines are slower—sometimes much slower—than routines that are optimized for a specific model. As a consequence, our earlier `SPost` commands (which we now refer to as `SPost9`), are actually much faster than the corresponding commands in `SPost13`. Interestingly, our `m*` commands take about as long to run today as our earlier commands took a decade ago. While those moving from `SPost9` to `SPost13` might be put off by how much slower the commands are, we think the advantages are overwhelming. With each new release of Stata, `margins` is faster, and your computer is likely to be more powerful. For now, however, in our own research, we take advantage of Stata 13 where `margins` is noticeably faster and use Stata/MP to take advantage of multiple computing cores.

The new methods of interpretation that are possible using `margins` and our `m*` commands sometimes require using loops and macros. They also require you to think carefully about how you want to compute predictions to interpret your model. We have marked some sections as *Advanced* to indicate that they involve more advanced methods of interpretation or require more sophisticated Stata programming. The box at the beginning of each of these sections explains what is being done in that section, why it is important, and when new users might want read the entire section. “Advanced” does not mean that the content of these sections is less valuable to some readers; indeed, we believe these sections include some of the most important contributions of this edition. Nor does it mean the commands are too difficult for substantive researchers. Rather, we think some readers might benefit from finishing the other sections in a chapter before reading the advanced sections.

We strongly encourage you to be at your computer so that you can experiment with the commands as you read. Initially, we suggest you replicate what is done in the book and then experiment with the commands. The `spost13.do` packages (see section 1.6.1) will download to your working directory the datasets we use and do-files to reproduce most of the results in the book. In the examples shown throughout the book, we assume the commands are being run in a working directory in which the `spost13.do` package has been installed. We have written the `spex` command (standing for “Stata postestimation examples”), which makes it simple to use the datasets and run the baseline estimation commands we use in the book. For example, the command `spex logit` downloads the data and fits the model we use as the baseline example for the binary logit model. After you type it, you are immediately ready to explore postestimation commands.

For each type of outcome that we consider in later chapters, we rely primarily on a single running example throughout. We have found this works best in teaching these materials. However, it does make selecting examples very challenging. We wanted examples to be interesting, accessible to diverse audiences, simple enough to follow easily without being trivial, representative of what you might find in other data, and illustrative of key points. In trying to balance these sometimes conflicting goals, we use examples that do not always make a compelling case for a particular method of interpretation. For example, an effect might be small or a plot rather uninteresting. We hope you will not decide on this basis that the method being illustrated will be ineffective with your data. A given method might not be effective in your application, but the nature of interpretation in nonlinear models is that you often need to try multiple approaches to interpretation until you find the one that is most effective. Of course, in some cases, the relationship you were expecting simply is not in the data you are analyzing. While we have tried dozens of variations and approaches to interpretation for each model, we cannot show them all. Just because we do not show a particular approach to interpretation for a given model does not imply that you should not consider that approach.

Conventions

We use several conventions throughout the book. Stata commands, variable names, filenames, and output are presented in a typewriter-style font like `this`. Italics are used to indicate that something should be substituted for the word in italics. For example, `logit variablelist` indicates that the command `logit` is to be followed by a list of variables.

When output from Stata is shown, the command you would type into Stata is preceded by a period (which is the Stata prompt). For example,

```
. logit lfp age wc hc k5, nolog
Logistic regression               Number of obs   =       753
(output omitted)
```


To reproduce the output, do not type the period before the command. If following along, only type those commands with a dot prompt. Commands without the dot prompt are just shown as examples. Also, as just illustrated, when we have deleted part of the output, we indicate this with *output omitted*. Keystrokes are set in **this font**. For example, Alt-f means that you are to hold down the Alt key while pressing the f key. The headings for sections that discuss advanced topics are indicated as **Advanced**.

We refer to entries in the Stata manuals by using the Stata convention in which the abbreviation for the manual is presented in brackets and the topic in boldface (for example, [R] **logit**). Typing `help manuals` in Stata will provide more information about these abbreviations.

The screenshots that we present are from Stata 13 for Windows. If you are using a different operating system or version, your screen might not look the same. See the StataCorp publication *Getting Started with Stata* for your operating system for further details.

The SPost commands

Many of the commands that we discuss are commands that we have written, and as such, they are not part of official Stata. To follow examples, you must install these commands as described in section 1.5. Although we assume you are using Stata 12 or later, most commands should work in Stata 11 (though we cannot support problems you might encounter in Stata 11).

Stata 13 added two features that we find extremely valuable. First, output for factor variables is much clearer. Second, it is possible to use `margins` to compute average discrete changes for variables that are not binary (full details are given in chapter 3).

The `spost13.ado` package cannot be installed along with the older `spost9.ado`. While the SPost9 commands `asprvalue`, `case2alt`, `misschk`, `mlogplot`, `mlogview`, `praccum`, `prchange`, `prcounts`, `prgen`, `prrtab`, and `prvalue` have been dropped from SPost13, their functionality remains in other commands. The `pr*` commands are replaced by the `m*` commands. `mlogplot` and `mlogview`, written using Stata 7 graphics and dialog boxes, have been replaced by `mlogitplot` and the related `mchangeplot` commands. `misschk` is no longer needed because of the introduction of Stata's `misstable` command. The `asprvalue` and `case2alt` commands have been mostly superseded by changes Stata has made to fitting models with alternative-specific variables.

Still, if you have used SPost9, you might want to use some of the old commands. With this in mind, we created the `spost9_legacy` package that includes commands that have been dropped. The versions of the commands in this package are not exactly the same as those in the `spost9.ado` package, but they have the same syntax as the earlier commands. We cannot, however, provide technical support for this package. If you need to run the SPost9 commands as described in the second edition—for example, to continue work on a project using these commands—you should uninstall `spost13.ado` and then install `spost9.ado`.

Getting help

We are gratified that many people have bought our book, but as a consequence, we receive many emails with questions. While we try to respond to everyone who contacts us, this is not always possible. Please read section 2.3 for information on the best way to resolve questions or problems as quickly as possible. We appreciate it.



THE
LIBRARY
OF THE
MUSEUM
OF
COMPARATIVE ZOOLOGY
AND ANATOMY
HARVARD UNIVERSITY
CAMBRIDGE, MASS.
U.S.A.
1954

1 Introduction

1.1 What is this book about?

Our book shows you effective and efficient ways to use regression models for categorical and count outcomes. It is a book about data analysis and is not a formal treatment of statistical models. To be effective in analyzing data, you want to spend your time thinking about substantive issues and not laboring to get your software to generate the results of interest. Accordingly, good data analysis requires good software and good technique.

Although we believe that these points apply to all data analyses, they are particularly important for the regression models that we examine. These models are nonlinear; consequently, the simple interpretations that are possible in linear models are no longer appropriate. In nonlinear models, the effect of each variable on the outcome depends on the level of all variables in the model. Because of this nonlinearity, which we discuss in detail in chapter 3, no method of interpretation can fully describe the relationships among the independent variables and the outcome. Rather, a series of postestimation explorations are needed to uncover the most important aspects of these relationships. If you limit your interpretations to the standard output of estimated slope coefficients, your interpretation will usually be incomplete and sometimes even misleading.

In the linear regression model (LRM), most of the work of interpretation is complete once the estimates are obtained. You simply read off the coefficients, which can be interpreted as follows: "For a unit increase in x_k , y is expected to increase by β_k units, holding all other variables constant." In nonlinear models, such as logit or negative binomial regression, additional computations are necessary after the estimates are obtained. Indeed, when interpreting nonlinear models, most of the work involves sometimes complex postestimation analyses, which are the focus of our book.

To make these computations, we use Stata's postestimation commands along with commands that we have written. Without these commands, the computations are time consuming and error-prone. All in all, it is not fun work, and it is tempting to limit your analyses to an uninformative table of parameter estimates. Fortunately, the commands we discuss in this book make sophisticated postestimation analysis routine and even enjoyable. Although these analyses can take a lot of work, our commands reduce the tedium so that you can focus on substantive issues.

1.2 Which models are considered?

Regression models analyze the relationship between an explanatory variable and an outcome variable while controlling for the effects of other variables. The LRM is probably the most commonly used regression model in the social sciences. A key advantage of the LRM is the ease of interpreting results. Unfortunately, this model applies only to cases in which the dependent variable is unbounded. Using the LRM when it is not appropriate may produce coefficients that are biased and inconsistent, and there is nothing advantageous about the simple interpretation of results that are incorrect. Fortunately, many appropriate models exist for categorical outcomes, and these models are the focus of our book. We cover models for four kinds of dependent variables: binary outcomes, ordinal outcomes, nominal outcomes, and count variables.

Binary outcomes have two values, such as whether a citizen voted in the last election, whether a patient was cured after receiving some medical treatment, or whether a respondent attended college. The regression models and commands we consider include binary logit (**logit**) and binary probit (**probit**).

Ordinal outcomes have more than two categories that are assumed to be ordered on a single, underlying dimension. For example, a survey might ask if you would be “very likely”, “somewhat likely”, or “not at all likely” to take a new subway to work, or if you agree with the president on “all issues”, “most issues”, “some issues”, or “almost no issues”. We focus on the ordered logit (**ologit**) and ordered probit (**oprobit**) models, but we also consider the sequential logit model (**seqlogit**), stereotype logistic regression (**slogit**), and the generalized ordered logit (**gologit2**), which is also appropriate for nominal outcomes.

Nominal outcomes also have more than two categories, but the categories are not ordered. Examples include the mode of transportation a person takes to work (for example, bus, car, train) or an individual’s employment status (for example, employed, unemployed, out of the labor force). The primary model we consider is the multinomial logit model (**mlogit**) along with its counterpart, the multinomial probit model with uncorrelated errors (**mprobit**). We also review the related conditional logit model (**clogit** and **asclogit**), the alternative-specific multinomial probit with correlated errors (**asmprobit**), and the rank-ordered logit model (**rologit**).

Finally, count variables count the number of times something has happened, such as the number of articles written by a scientist or the number of patents a biotechnology company has obtained. We begin with the Poisson regression model (**poisson**), followed by the negative binomial regression model (**nbreg**), the zero-truncated Poisson and negative binomial models (**tpoisson** and **tnbreg**), the hurdle regression model, and lastly, the zero-inflated Poisson and negative binomial models (**zip** and **zinb**).

Although this book covers many models for different types of outcomes, they are all models for cross-sectional data. We do not consider models for survival or event-history data, even though Stata has a powerful set of commands for dealing with these data. We recommend Cleves et al. (2010) and the *Stata Survival Analysis Reference Man-*

ual for more information on these types of models. Likewise, we do not consider any models for panel or other multilevel data, even though Stata contains commands for fitting these models. For additional information, see Rabe-Hesketh and Skrondal (2012), Cameron and Trivedi (2010), and the *Stata Longitudinal-Data/Panel-Data Reference Manual*.

1.3 Whom is this book for?

We expect that readers of this book will vary considerably in their knowledge of both statistics and Stata. With this in mind, we have tried to structure the book to accommodate the diversity of our audience. Minimally, however, we assume that you have a solid familiarity with the linear regression model and that you are comfortable using the basic features of the operating system of your computer. Although we have provided sufficient information about each model so that you can read each chapter without prior exposure to the models discussed, we strongly recommend that you do not use this book as your sole source of information on the models (see section 1.8 for reading recommendations). Our book will be most useful if you have already studied or are studying the models considered herein in conjunction with reading our book.

Ideally, you are running Stata 13 or later. Most of our examples will, however, run in Stata 11 and 12. If you are using a version of Stata earlier than Stata 11, we suggest that you instead use the second edition of our book (Long and Freese 2006). However, with the powerful new features in Stata 13 and the new methods of interpretation in this third edition, we hope you decide instead to upgrade your software. To make the most out of the book, you will need access to the Internet to download our commands, datasets, and sample programs (see section 1.5 for details). For information about obtaining Stata, see the StataCorp website at <http://www.stata.com>.

1.4 How is the book organized?

Chapters 2–4 introduce materials that are essential for working with the models we present in the later chapters:

Chapter 2: Introduction to Stata reviews the basic features of Stata that are necessary to get new or inexperienced users up and running with the program. New users should work through the brief tutorial that we provide in section 2.18. This introduction is by no means comprehensive, so we include information on how to learn more about using Stata. Those who are familiar with Stata can skip this chapter, although even these readers might benefit from scanning it.

Chapter 3: Estimation, testing, and fit reviews Stata commands for fitting models, testing hypotheses, and computing measures of model fit. Those who regularly use Stata for regression modeling might be familiar with much of this material; however, we suggest at least a quick review of the material. Most importantly,

you should read our detailed discussion of factor-variable notation, which was introduced in Stata 11. Understanding how to use factor variables is essential for the methods of interpretation presented in the later chapters.

Chapter 4: Methods of interpretation is an overview of various approaches to interpreting regression models. We introduce the `margins` command that is part of official Stata and the `mtable`, `mgen`, and `mchange` commands that are part of `SPost13`. This chapter is essential background before proceeding to part II. Study this chapter carefully, even if you are an advanced user. Readers new to Stata are likely to find that this chapter has more detail than initially needed; therefore, throughout the chapter, we suggest which sections you may wish to only skim on first reading.

Part II covers regression models for different types of outcomes.

Chapters 5 and 6: Models for binary outcomes begin with an overview of how the binary logit and probit models are derived and how they can be fit. After the model has been fit, we show how to test hypotheses, compute residuals and influence statistics, and calculate scalar measures of model fit. Chapter 6 uses postestimation commands that assist in interpretation using predicted probabilities, discrete and marginal change in the predicted probabilities, and for the logit model, odds ratios. Because binary models provide a foundation on which many models for other kinds of outcomes are derived, and because these two chapters provide more detailed explanations of common tasks than later chapters do, we recommend reading these chapters carefully even if you are interested mainly in another type of outcome.

Chapter 7: Models for ordinal outcomes presents the ordered logit and ordered probit models. We show how these models are fit and how to test hypotheses about coefficients. We also consider tests of a key assumption of both models, known as the parallel regression assumption. For interpreting results, we discuss methods similar to those described in chapter 6, and we also discuss interpretation in terms of a latent dependent variable. Methods of interpretation using predicted probabilities apply directly to models for nominal outcomes, so it is useful to familiarize yourself with these methods before proceeding to chapter 8. This chapter also details the implications of assuming that an ordinal model is appropriate for your outcome and recommends that you use models for nominal outcomes as part of your evaluation of ordinal models.

Chapter 8: Models for nominal outcomes focuses on the multinomial logit model. We show how to test hypotheses that involve multiple coefficients and discuss tests of a key assumption known as the independence of irrelevant alternatives assumption. Methods of interpretation using predictions are identical to those for ordinal models. Interpretation using odds ratios is a simple extension of the methods introduced in chapter 6, although the multinomial logit model's many parameters make the process of interpretation much more complicated. To deal with

this complexity, we present a graphical method for summarizing the parameters. The multinomial probit model without correlated errors is discussed briefly, and then the multinomial logit model is used to explain the stereotype logit model. This model, which is often used with ordinal outcomes, also has applications with nominal outcomes. These models assume case-specific independent variables (each independent variable has one value for each observation). We end the chapter with a short review of models that also include alternative-specific data, in which some variables vary over the alternatives for each individual, such as an individual's similarity to each candidate in an election. We consider the conditional logit model and the alternative-specific multinomial probit model, the latter of which allows correlations between alternative-specific error terms. Lastly, we present the rank-ordered logistic regression model, which can be used when you have information about the ranking of outcomes as opposed to information about only the selected or most preferred outcome.

Chapter 9: Models for count outcomes begins with the Poisson and negative binomial regression models, including a test to determine which model is appropriate for your data. We also show how to incorporate differences in exposure time into parameter estimation. Next, we consider interpretation for changes in the predicted rate and changes in the predicted probability of observing a given count. The rest of the chapter deals with models that address problems associated with having too many zeros relative to what the model predicts or having no zeros at all. We start with zero-truncated models for which zeros are missing from the outcome variable, perhaps because of the way the data were collected. We then merge a binary model and a zero-truncated model to create the hurdle model. We also consider fitting and interpreting zero-inflated count models, which are designed to account for the many zero counts often found in count outcomes.

1.5 The SPost software

From our point of view, one of the best things about Stata is how easy it is to add your own commands. If Stata does not have a command you need or some command does not work the way you like, you can program a new command yourself, and it will work as if it were part of official Stata. We have created a suite of programs, referred to collectively as SPost13 (Stata postestimation commands for version 13), for the postestimation interpretation of regression models. These commands must be installed before you can try the examples in later chapters.

If you have used SPost before, read this! For this book, we completely rewrote our earlier SPost commands, which we will refer to as SPost9. If you have the `spost9.ado` package installed on your computer, you should uninstall it (details below) before you install the `spost13.ado` package.

To get the most out of this book, you need to try each method using both the official Stata commands and our *SPost13* commands. Ideally, you are running Stata 13 or later. If you are running Stata 11 or Stata 12, most of our examples will work, but a few valuable features new in Stata 13 will not be available. Before we discuss how to install our commands and update your software, we have suggestions for new Stata users, those with earlier versions of Stata, and those who used *SPost9*:

If you are new to Stata. If you have never used Stata, you might find the instructions in this section to be confusing. It might be easier if you skim the material now and return to it after you have read the introduction to Stata in chapter 2.

If you are using Stata 10 or earlier. The *SPost13* commands used in this book will not run with Stata 10 and earlier. You can use *SPost9* contained in the `spost9.ado` package, which is described in the second edition of our book (Long and Freese 2006). However, if you are investing the time to learn these methods, we think you are much better off upgrading your software so that you can use `spost13.ado`.

If you are using an earlier version of *SPost*. Before using the `spost13.ado` package, you must uninstall any earlier versions of *SPost*, such as the `spostado` package or the `spost9.ado` package. *SPost13* replaces our earlier `prvalue`, `prtab`, `prgen`, `prchange`, and `praccum` commands with the more powerful `mtable`, `mgen`, `mchange`, and `mlistat` commands based on Stata's remarkable `margins` command, which did not exist when the previous edition of this book was written. If you want to use our new commands but also want access to the *SPost9* commands, you can install the `spost9_legacy` package. Details are given below.

1.5.1 Updating Stata

Before you install *SPost13*, we strongly recommend that you update your version of Stata. This does not mean to upgrade to Stata 13, but rather to make sure you have the latest updates for whatever version of Stata you are running. You should do this even if you have just installed Stata because the DVD or download that you received might not have the latest changes to the program. When you are online and in Stata, you can update Stata by selecting **Check for Updates** from the **Help** menu; equivalently, you can type the command `update query`, as we did here:



This screen shows the update status and recommends an action. Our installation is up to date. If it was not, Stata would recommend an update and would provide instructions on how to do that.

1.5.2 Installing SPost13

We begin with some background of what happens when you install user-written commands in Stata. The good news is that once they are installed, these commands behave just like official Stata commands. Programs that add commands to Stata are contained in files that end in the extension `.ado`, which stands for automatic do-file. For example, `listcoef.ado` is the file that contains the command `listcoef`. When you type the command `listcoef`, Stata automatically runs `listcoef.ado`. The `ado`-files, along with supplementary files that might have other suffixes, are included as part of a package. In Stata, a package comprises a list of included files along with instructions on how to install them on your computer.

To install SPost13, you install the `spost13.ado` package. We consider two methods for installation, but first we explain how to uninstall the `spost9.ado` package.

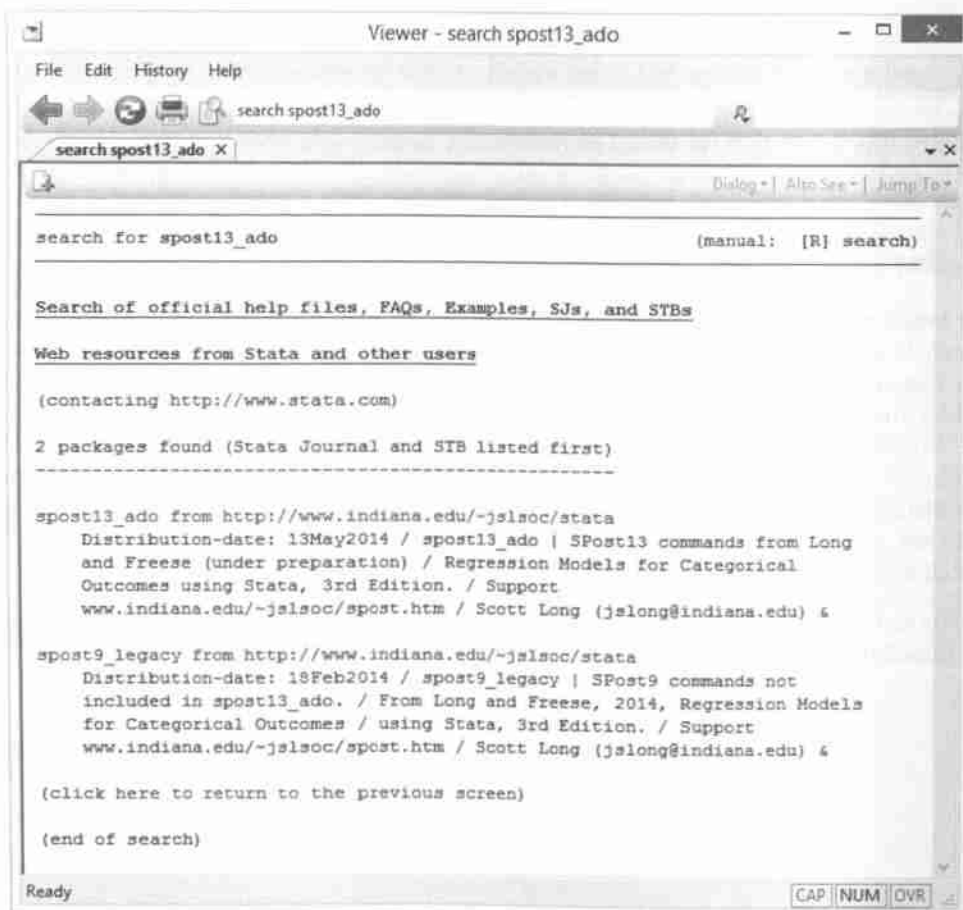
Uninstalling SPost9

To determine if you have this package installed, type the command `ado`. If `spost9_ado` is listed, you can uninstall it by typing

```
. ado uninstall spost9_ado
package spost9_ado from http://www.indiana.edu/~jlsoc/stata
Distribution-date: 25Jul2012
(package uninstalled)
```

Installing SPost13 using search

The `search word` command searches an online database wherein StataCorp keeps track of user-written additions to Stata. Typing `search spost13_ado` opens a Viewer window that looks like this:



When you click on `spost13_ado` from <http://www.indiana.edu/~jslsoc/stata>, which is a blue link, a new Viewer window opens:



Click on the linked text that says (click here to install). After a delay during which files are downloaded, Stata will respond with one of the following messages:

installation complete means that the package has been successfully installed and that you can now use the commands. Just above the **installation complete** message, Stata tells you the directory where the files were installed.

all files already exist and are up to date means that your system already has the latest version of the package. You can now use the commands.

the following files exist and are different indicates that your system already has files with the same names as those in the package being installed and that the existing files differ from those in the package. The names of the differing files are listed, and you are given several options. Most likely, the files listed are earlier versions of our programs, so you should select the option **Force installation replacing already-installed files**. This is not as ominous as it sounds. Because the files on our website are the latest versions, you want Stata to replace your current files with these new files.

cannot write in directory *directoryname* means that you do not have write privileges to the directory where Stata is trying to install the files. This usually occurs only when you are using Stata on a network. We recommend that you contact your network administrator and ask whether our commands can be installed using the instructions given above. If you cannot wait for a network administrator to install the commands or to give you the needed write access, you can install the programs to any directory where you do have write permission, including a flash drive or your personal directory on a network. For example, suppose that you want to install SPost13 to your directory called *d:\username* (which can be any directory where you have write access). You should use the following commands:

```
. cd d:\username
d:\username
. mkdir ado
. sysdir set PERSONAL "d:\username\ado"
. net set ado PERSONAL
. net search spost
(contacting http://www.stata.com)
```

Then follow the installation instructions provided above for installing SPost13. If you get the error "could not create directory" after typing `mkdir ado`, you probably do not have write privileges to the directory.

If you install ado-files to your own directory, then each time you begin a new Stata session you must tell Stata where these files are located. You do this by typing `sysdir set PERSONAL directoryname`, where *directoryname* is the location of the ado-files. For example,

```
. sysdir set PERSONAL d:\username\ado
```

Installing SPost13 using net install

You can also install the `spost13.ado` package entirely from the Command window. If the method listed above does not work, the following steps might. While online, type

```
. net from http://www.indiana.edu/~jslsoc/stata/
```

Available packages will be listed in the Results window. You can click on `spost13.ado` and follow the instructions, or you can type


```
. net install spost13_ado
```

`net get` can be used to download supplementary files (for example, datasets and sample do-files) from our website. For example, to download the package `spost13.do` (discussed below), type

```
. net get spost13.do
```

These files are downloaded to the current working directory (see chapter 2 for a full discussion of the working directory).

1.5.3 Uninstalling SPost13

If you want to uninstall our commands, simply type `ado uninstall spost13.ado`. When things do not seem to work right, our first suggestion is to uninstall `spost13.ado` and then reinstall it.

1.6 Sample do-files and datasets

Although we hope you will soon be using the methods in the book with your own data, we think it is valuable to first reproduce our examples and then modify them to try new things. To facilitate this, we have written the `spex` command (included in `spost13.ado`), which makes it easy to load our data and run our baseline models. We also created the `spost13.do` package, which lets you download the data and do-files for each chapter.

1.6.1 Installing the `spost13.do` package

The `spost13.do` package contains the datasets used in the book along with do-files that reproduce most of the analyses. The do-files have names like `rm3ch9-count.do` and contain comments that link the commands to sections of the book. To download these files to your working directory, while in Stata and online, type `search spost13.do`. In the Viewer window that opens, click on the blue link `spost13.do` from <http://www.indiana.edu/~jslsoc/stata>. A new Viewer window will open, and you can follow the instructions to download the files.

In the examples shown throughout the book, we assume the commands are being run from within a working directory in which the `spost13.do` package has been installed. These do-files assume that you are using Stata 13.1. If you are using Stata 12, install the `spost13.do13` package instead.

1.6.2 Using *spex* to load data and run examples

Experimenting with the postestimation commands that we discuss requires that you have first fit the appropriate model. In our examples, we show you how to open a dataset and fit models as you would if you were working with your own data. Accordingly, we

begin with a `use` command to load the data and then use an estimation command, such as `logit`, to fit the model.

To make it simpler for you to experiment with the methods in later chapters, we have written the command `spex` (Stata postestimation examples). Typing `spex commandname` will produce our primary example for that estimation command. If you type `spex logit`, for example, Stata will automatically load the data and fit the model that serves as our main logit example. Alternatively, you can specify the name of any dataset that we use (`spex datasetname`), and `spex` will load those data but not fit any model. By default, `spex` looks for the dataset on our website. If it does not find the dataset there, it will look in the current working directory and all the directories where Stata searches for ado-files. For more information, type `help spex`.

The running examples in this edition of the book are different from those used in the second edition. If you want to run the earlier examples, use the `spex9` command. For example, `spex9 logit` runs the logit example that was used with `spost9_ado`.

1.7 Getting help with SPost

Because things do not always work as intended and commands that we say will work might not, we have some troubleshooting recommendations for you. We ask you to please read this section carefully to try to resolve any problems you may be experiencing with SPost13. If none of these suggestions fixes the issue, you can then contact us.

1.7.1 What if an SPost command does not work?

We assume here that you have installed SPost13 but that some of or all the commands do not work. Here are some things to consider:

1. Make sure Stata is properly installed and up to date. Typing `verinst` will verify that Stata has been properly installed. Typing `update query` will tell you whether the version you are running is up to date and what you should do next. If you are running Stata over a network, your network administrator may need to do this for you. See [U] 28 Using the Internet to keep up to date and [R] `update`.
2. Make sure SPost13 is up to date. Type `adoupdate`, `update` to check, or uninstall `spost13_ado` and then reinstall it.
3. If you get the error message `unrecognized command`, there are several possibilities.
 - a. If the command used to work, consider whether you are working on a different computer or station in a computer lab. User-written programs must be installed on each machine that you use.
 - b. If you sent a do-file using SPost13 commands to another person who cannot get the commands to work, that person should verify he or she has SPost13 installed. Your do-file will not work with SPost9.

- c. If you get the error message `unrecognized command: strangename` after typing one of our commands, where *strangename* is not the name of the command that you typed, it means that Stata cannot find an ancillary ado-file that the command needs. We recommend that you uninstall the `spost13_ado` package and then reinstall it.
4. If you get an error message that you do not understand, click on the blue return code beneath the error message for more information about the error.
5. Often, what appears to be a problem with one of our commands is actually a mistake the user has made. We know this because we make these mistakes, too. For example, make sure that you are not using `=` when you should be using `==`.
6. Because our commands are for use after you have fit a model, they will not have the information needed to operate properly if Stata was not successful in fitting your model. Our commands should trap such errors but sometimes do not, so make sure there were no problems with the last model fit.
7. Irregular value labels can cause commands to fail. Where possible, we recommend using labels that have fewer than eight characters and contain no spaces or special characters other than underscores (`_`). If you are having problems and your variables do not meet this standard (especially the labels for your dependent variable), then try changing your value labels with the `label` command (details are given in section 2.15).
8. Unusual values of the outcome categories can also cause problems. For ordinal or nominal outcomes, some of our commands require that all the outcome values be integers between 0 and 99. The behavior of some official Stata commands can also be confusing when unusual values are used. For these types of outcomes, we strongly recommend using consecutive integers starting with 1.

1.7.2 Getting help from the authors

If you have tried everything we recommended in section 1.7.1 and you are still encountering an error, the next step is to contact us. We hear from hundreds of readers and do our best to help. To make this easier for us, please carefully follow the suggestions in this section.

We encourage you to start by reviewing William Gould's blog entry "How to successfully ask a question on Statalist" (2010). His advice will increase your chances of getting your question answered, either from the Statalist, from us, or elsewhere. In addition, we have found that in the process of carefully preparing a question, we often find the solution ourselves.

Here are other suggestions to make it easier for us to answer your question, which will also increase your chances of getting a prompt answer:

1. Check <http://www.indiana.edu/~jslsoc/spost.htm> and http://www.indiana.edu/~jslsoc/spost_help.htm for advice on what to try before contacting us. There might be recent information that solves your problem.
2. Make sure that both Stata and `spost13.ado` are up to date. We keep repeating this because it is the most common solution to problems our readers bring to us.
3. Look at the sample files in the `spost13.do` package. It is sometimes easiest to figure out how to use a command by seeing how others use it. Try what you are doing on one of our datasets and see if you can reproduce it.
4. If you still have a problem, send us the information described in the next section.

What we need to help you

To solve your problem, we need to be able to reproduce it. Simply describing the problem rarely is sufficient. Please send us the following:

1. A do-file that reproduces the problem (see a sample do-file at the end of this section).
 - a. Include your name, email address, and a description of the problem.
 - b. Begin with the commands `about` and `spost13which`, which displays the versions of software you are using.
 - c. Include the results of `summarize` used in the analysis (not all variables in the dataset) and the results of `tabulate` for categorical variables.
 - d. Include only the commands needed to reproduce or explain the problem. Remove all unrelated commands.
 - e. Remove all references to specific directories, such as `log using c:\data\project3\problem`, text or use `c:\data\project3\sample.dta`. Our computer will not have your directory structure, so your do-file will not run on our computer. The do-file should read the data from the working directory and save the log file to the working directory.
2. The dataset used by the do-file. This should be a small dataset extracted from the full dataset you are using. Only send the variables used in your do-file and create a dataset with only a subset of your observations (assuming, of course, that the error is reproduced with the smaller sample).
3. A plain text log file showing the error. Do not send the log in SMCL format. To avoid this, add the `text` option to your `log` command, for example, `log using myproblem.log, text replace`.

Send this information to jslong@indiana.edu or jfreese@northwestern.edu.

Here is an example of what a do-file might look like:

```
capture log close
log using yourname.log, text replace
* mtable generates a variable not found error.
* scott long - jslong@indiana.edu - 2014-05-09
about
spost13which
use jslong-error.dta
logit y x1 x2
* the following command causes the error
mtable x1, at(x2=(1(1)3))
log close
```

1.8 Where can I learn more about the models?

If you want to learn more about the regression models that are covered in this book, we recommend the following valuable sources:

- Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press. This is an excellent introduction to the methods and models discussed in this book, as well as models for panel data.
- Cameron, A. C., and P. K. Trivedi. 2010. *Microeconometrics Using Stata*. Rev. ed. College Station, TX: Stata Press. This companion to *Microeconometrics: Methods and Applications* (Cameron and Trivedi 2005) shows how to use Stata for cross-section and panel models.
- Cameron, A. C., and P. K. Trivedi. 2013. *Regression Analysis of Count Data*. 2nd ed. Cambridge: Cambridge University Press. This is a definitive reference about count models.
- Hardin, J. W., and J. M. Hilbe. 2012. *Generalized Linear Models and Extensions*. 3rd ed. College Station, TX: Stata Press. This is a thorough review of the generalized linear model (or GLM) approach to modeling and includes detailed information about using these models with Stata.
- Hosmer, D. W., Jr., S. Lemeshow, and R. X. Sturdivant. 2013. *Applied Logistic Regression*. 3rd ed. New York: Wiley. This book, written primarily for biostatisticians and medical researchers, considers logit models for binary, ordinal, and nominal outcomes. The authors often discuss how their recommendations can be executed using Stata.
- Long, J. Scott. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage. This book provides more details about the models discussed in our book.

- Train, K. 2009. *Discrete Choice Methods with Simulation*. 2nd ed. New York: Cambridge University Press. This is a thorough review of a wide range of models for discrete choice and includes details on new methods of estimation using simulation.
- Wooldridge, J. M. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press. This is a comprehensive review of econometric methods for cross-section and panel data.

2 Introduction to Stata

This book is about fitting and interpreting regression models using Stata; to earn our pay, we must get to these tasks quickly. With that in mind, this chapter is a relatively concise introduction to Stata 13 for those with little or no familiarity with the software. Experienced Stata users can skip this chapter, although a quick reading might be useful. We focus on teaching the reader what is necessary to work through the examples later in the book and to develop good working techniques when using Stata for data analysis. These discussions are not exhaustive; often, we show you either our favorite approach or the approach that we think is simplest. One of the great things about Stata is that there are usually several ways to accomplish the same thing—so if you find a better way than what we have shown you, use it!

You cannot learn how to use Stata simply by reading. We strongly encourage you to try the commands as we introduce them. We have also included a tutorial in section 2.18 that covers the basics of using Stata. Indeed, you might want to try the tutorial first and then read our detailed discussions of the commands.

The screenshots in this chapter were created in Stata 13.1 running under Windows using the default windowing preferences. If you have changed the defaults or are running Stata under Unix or Mac OS, your screen might look slightly different. Those of you new to Stata, regardless of the operating system you are using, should examine the appropriate *Getting Started* manual, available in PDF format with your copy of Stata, for further details: *Getting Started with Stata for Mac*, *Getting Started with Stata for Unix*, or *Getting Started with Stata for Windows*. How to access this and other documentation from within Stata is discussed in section 2.3.2.

For further instruction beyond what is provided in this chapter, look at the resources listed in section 2.3. We assume that you know how to load Stata on the computer you are using and that you are familiar with your computer's operating system. By this, we mean that you should be comfortable copying and renaming files, working with subdirectories, closing and resizing windows, selecting options with menus and dialog boxes, and so on.

2.1 The Stata interface

Figure 2.1 shows what Stata looks like after several commands have been entered and data have been loaded into memory. The five main windows that you will use most often are the Review, Results, Command, Variables, and Properties windows. Except

for the large Results window, each has its name listed in its title bar. There are also other, more specialized windows, such as the Viewer, Data Editor, Variables Manager, Do-file Editor, Graph, and Graph Editor windows.

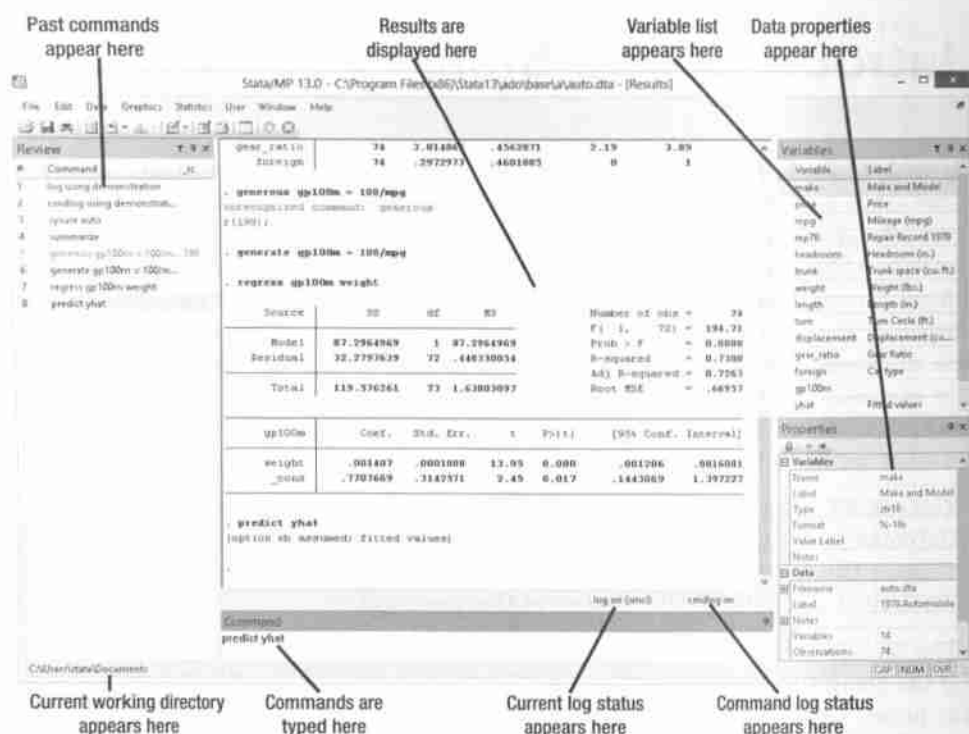


Figure 2.1. The Stata user interface

The **Command window** is where you type commands that are executed when you press Enter. As you type commands, you can edit them at any time before pressing Enter. Pressing Page Up brings the most recently used command into the Command window, where you can edit it and then press Enter to run the modified command.

The **Results window** echoes the command typed in the Command window (the commands are preceded by a ".", called the dot prompt, as shown in figure 2.1) and then displays the output from that command. Within the window, you can highlight text and right-click on that text to see options for copying the highlighted text. The **Copy Table** option copies the selected lines to the Clipboard, and **Copy Table as HTML** allows you to copy the selected text as an HTML table (see page 114 for more information). You also have the option to print the contents of the window. Only the most recent output is available this way; earlier


lines are lost unless you have saved them to a log file (discussed below). Details on setting the size of the scrollbar buffer are given below.

The Review window lists the commands that have been typed in the Command window. If you click on a command in this window, it is pasted into the Command window, where you can edit it and then press **Enter** to run the modified command. If you double-click on a command in this window, the command is immediately executed.

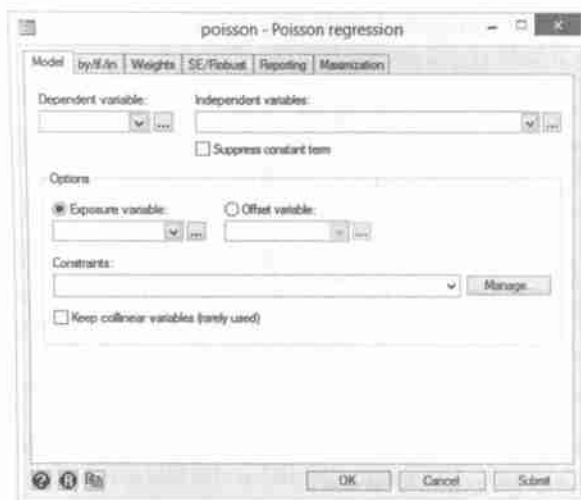
The Variables window lists the names and labels of all the variables for the dataset in memory. If you click on a variable in this window, information about it is shown in the Properties window. If you double-click on a variable in this window, its name is pasted into the Command window.

The Properties window lists attributes of the most recently selected variable and of the dataset as a whole. For example, under the Data tab, you can see at a glance how many variables and observations are in your dataset.

The Command and Results windows illustrate Stata's origins in a command-based system. That is, you tell Stata what to do by typing commands that consist of one line of text and then pressing **Enter**. At the same time, there is a graphical user interface (GUI) for accessing virtually all commands. At the risk of seeming old-fashioned, however, we greatly prefer the command-based interface. Although it can take longer to learn, you will find it much faster to use once you do learn it. If you currently prefer using pull-down menus, stick with us, and you will likely change your mind. Also, although we first consider entering only one command at a time, in section 2.9, we show you how to run a series of commands at once, which is vital to doing serious work in Stata and requires moving beyond the GUI to do-files.

This is not to say that we never use the mouse when using Stata. For example, some important tasks can be performed by clicking on icons on the toolbar at the top of the screen. For example, if you click on the **Data Editor (Browse)** button, , Stata opens a spreadsheet for examining your data. Instead of clicking on the icon, you could have done the same thing by typing **browse** in the Command window.

You can also use menus and dialogs to create commands. At the top of figure 2.1 are a series of menus, beginning with **File**, then **Edit**, and so on. The **Data**, **Graphics**, and **Statistics** menus provide point-and-click access to almost every command in Stata. For example, instead of typing the **poisson** command for the Poisson regression model that is discussed in chapter 9, you could select **Statistics > Count outcomes > Poisson regression**, which displays this dialog:



This dialog box gives you access to all the options for the `poisson` command and allows you to select variables. After you construct your command in the dialog box, you click on **Submit** to send the command to the Command window and execute it. We find this feature especially useful for making graphs, because the options for making graphs are many and can be hard to remember. If we use dialog boxes to make a good approximation of the graph we want, Stata will put the syntax in the Results window. We can then copy and tweak the syntax to get our graph just right. We illustrate doing this below.

We urge you to get used to working in Stata by entering commands. It is ultimately much faster. It also makes things much easier to automate later, which is key to doing work that you can reproduce and modify in the future because you have a complete record of the commands used to create your results. Consequently, we describe things below mainly in terms of commands.

That said, we also encourage you to explore the tasks available through menus and the toolbar and to figure out what combination of dialogs and commands works most easily and efficiently for you.

Changing the scrollbar buffer size

How far back you can scroll in the Results window is controlled by the command

```
set scrollbufsize #
```

where $10,000 \leq \# \leq 2,000,000$. By default, the buffer size is 200,000 bytes. When you change the size of the buffer using `set scrollbufsize`, the change will take effect the next time you launch Stata. This means that if you are trying to look at earlier results and realize your scroll buffer setting is too small, you will have to relaunch Stata and rerun your analyses. Unless computer memory is a problem, we recommend you set the buffer to its maximum.

Tip: Changing defaults. If you type `help set`, you can get an idea of the range of different parameters that users may set in Stata. Many of these can be switched permanently by adding the `permanently` option. For preferences that cannot be set permanently in this way, you could enter the command to reset the parameter at the start of each Stata session, but it is easier to add all the commands to reset parameters to `profile.do`, a file that is automatically run each time Stata begins. See [GS] **B.3 Executing commands every time Stata is started** for details.

2.2 Abbreviations

Commands and variable names often can be abbreviated. For variable names, the rule is easy: Any variable name can be abbreviated to the shortest string that uniquely identifies it. For example, if there are no other variables in memory that begin with `a`, the variable `age` can be abbreviated as `a` or `ag`. If you have the variables `income` and `income2` in your data, neither of these variable names can be abbreviated.

There is no general rule for abbreviating commands, but as you might expect, typically the most common and general command names can be abbreviated. For example, four of the most often used commands are `summarize`, `tabulate`, `generate`, and `regress`, and these can be abbreviated as `su`, `ta`, `g`, and `reg`, respectively. Although very short abbreviations are easy to type, they can be confusing when you are getting started. Accordingly, when we use abbreviations, we stick with at least three-letter abbreviations.

2.3 Getting help

We briefly review here the many ways to get help as you use Stata. If you need more information about getting help, type the command `help help`, read the information that appears in a Viewer window, and click on anything shown in blue type that sounds interesting. (Text shown in blue in the Viewer or Results window is a link to additional information.)

2.3.1 Online help

If you find our description of a command incomplete, or if we use a command that is not explained, you can use Stata to find more information. Use the `help` command when you know the name of the command you want more information about. For example, `help regress` pulls up information about the `regress` command. The `search` command is more general: you are given a list of references related to your search. For example, `search regress` lists over 100 references with information about “regress” in Stata manuals, the Stata website (including frequently asked questions, or FAQs), and articles

from the *Stata Journal* (often abbreviated SJ). **search** even provides information about related user-written commands that are not part of official Stata. The **search** command is so useful for tracking down information that we encourage you to type **help search** and read more about how it works. Alternatively, see [GS] 4 **Getting help** in the Stata documentation.

2.3.2 PDF manuals

The Stata manuals are extensive, and it is worth taking an hour to browse them to get an idea of the many features available in Stata. The manuals come as PDF files that you can access from within Stata through the menu system. Select the menu item **Help**→**PDF Documentation** to open a list of all the Stata manuals, which you can then browse individually by clicking on the one you want to open. You can also easily access a specific manual entry from its help file. Suppose you type the command **help regress**. In the Viewer window that opens, you will see in blue [R] **regress** -- **Linear regression**. If you click on this link, a PDF of the *Stata Base Reference Manual* will open to the entry for the **regress** command. Within the manuals, you will find many cross-references in the form of clickable links that allow you to browse easily among related topics. See [GS] 4 **Getting help** and [U] 1.2 **The User's Guide and the Reference manuals** for further details.

In general, we find that learning how to read the manuals and use the help system is more efficient than asking someone, and it allows you to save your questions for the really hard stuff. For those new to Stata, we recommend Stata's *Getting Started* manual (which is specific to your platform) and the first part of the *Stata User's Guide*. As you become more acquainted with Stata, the manuals will become increasingly valuable for detailed information about each command, including a discussion of the statistical theory related to the command and references for further reading.

2.3.3 Error messages

If you type an incorrect command, an error message appears in the Results window. The message is printed in red, along with a return code (for example, **r(199)**) listed in blue. Clicking on the return code provides a more detailed description of the error. Although error messages often can be helpful in resolving a problem, sometimes they are terse and even misleading. Stata knows how to understand a correct command but does not necessarily know what an incorrect command is trying to do. For additional information on debugging Stata programs, see Long (2009).

2.3.4 Asking for help

Sometimes, despite your best efforts, your program still will not work. Before you ask someone for help, take a few minutes to review William Gould's blog entry "How to successfully ask a question on Statalist" (2010). His advice will increase your chances

of getting your question answered, either from the Statalist, from us, or elsewhere. In addition, we have found that in the process of carefully preparing a question, we often find the solution ourselves. With this in mind, if you have questions about this book or the `SPost` commands, we suggest that you carefully read section 1.7 before contacting us. Thank you.

2.3.5 Other resources

The Stata website (<http://www.stata.com>) contains useful resources, including links to tutorials, an extensive FAQ section that discusses both introductory and advanced topics, and information about NetCourses and short courses. StataCorp has free tutorial videos on their YouTube channel (<http://www.youtube.com/user/statacorp>) that cover a variety of topics of interest. Another excellent resource is UCLA's Institute for Digital Research and Education at <http://www.ats.ucla.edu/stat/stata/>.

A Stata forum known as Statalist is not run by StataCorp, but many programmers and statisticians from StataCorp participate. This forum (<http://www.statalist.org/>) is a wonderful resource for information on Stata and statistics. You will often find that your questions have already been asked and answered by someone in the past, but you can submit your questions and often receive answers quickly. Monitoring the forum is also a good way to pick up insights from Stata veterans.

We also recommend Long's (2009) book on managing the workflow of data analysis projects. Chapter 3 of that book has a detailed discussion of how to use do-files effectively to efficiently produce results that can be reproduced. Chapter 4 extends the discussion of Stata automation that we only touch upon in our book.

2.4 The working directory

The working directory is the default directory for any file operations, such as using data, saving data, or logging output. If you type `cd` or `pwd` in the Command window, Stata displays the name of the current working directory (see `help cd` or `[D] cd`). To load a data file stored in the working directory, you simply type `use filename` (for example, `use binlfp4`). If a file is not in the working directory, you must specify the full path (for example, `use d:\spostdata\examples\binlfp4`). At the beginning of each Stata session, we like to change our working directory to the directory where we plan to work, because this is easier than repeatedly entering the path name for the directory. For example, typing `cd d:\spostdata` changes the working directory to `d:\spostdata`. If the directory name includes spaces, you must put the path in quotation marks (for example, `cd "d:\my work"`).

Stata's menu system allows you to change the working directory from the **File** menu. As always, when you use the menus to do this, the Stata command will appear in the Results window. You can then copy and paste this command into your do-files (described below) to automate the process. Stata also allows you to select a file from

your computer via **File→Filename...**; selecting a file this way simply pastes the full directory path and filename into the Command window for you so that you can easily copy it.

You can list the files in your working directory by typing `dir` or `ls`. With these commands, you can use the `*` wildcard. For example, `dir *.dta` lists all files with the extension `.dta` and so is a quick way to obtain a list of all the Stata data files in your working directory.

As we mentioned in chapter 1, the commands used in the examples in this book are available in the package `spost13.do`, which you can find and download by typing `search spost13`. When working through our examples, we do not specify a path when opening data files, because we assume those data files are already in your working directory.

2.5 Stata file types

Stata uses and creates many types of files, which are distinguished by extensions at the end of the filename. A full list of file extensions used in Stata is available by typing `help extensions`. The extensions you are likely to encounter are the following:

- `.ado` Programs that add commands to Stata, such as the `SPost` commands.
- `.do` Script or batch files that contain Stata commands.
- `.dta` Data files in Stata's format.
- `.gph` Graphs saved in Stata's proprietary format.
- `.log` Output saved as plain text by the `log using` command.
- `.smcl` Output saved in the SMCL format by the `log using` command.

The most important of these for a new user are the `.smcl`, `.log`, `.dta`, and `.do` files, all of which we will cover in the sections that follow.

2.6 Saving output to log files

Stata does not automatically save the output from your commands, it only shows the output in the Results window. To save your output to print or examine later, you must open a log file. Once a log file is opened, both the commands and the output they generate are saved in that log file. Because the commands are recorded, you can tell exactly how the results were obtained. The syntax for the `log` command is

```
log using filename [, append replace [smcl|text]]
```

By default, the log file is saved to your working directory. You can save it to a different directory by typing the full path (for example, `log using d:\project\mylog, replace`).

Options

append specifies that if the file exists, new output should be added to the end of the existing file.

replace indicates that you want to replace the log file if it already exists. For example, **log using mylog** creates the file **mylog.smcl**. If this file already exists, Stata generates an error message. So, you could use **log using mylog, replace**, and the existing file would be overwritten by the new output.

smcl and **text** specify the format in which the log is to be recorded.

smcl, the default option, requests that the log be written using the Stata Markup and Control Language (SMCL) with the file suffix **.smcl**. SMCL files contain special codes that add solid horizontal and vertical lines, bold and italic typefaces, and hyperlinks to the Results window. The disadvantage of SMCL is that the special features can be viewed only within Stata. If you open a SMCL file in a text editor, your results will appear amidst a jumble of special codes.

text specifies that the log be saved as plain text (ASCII), which is the preferred format for loading the log into a text editor for printing. Instead of adding the **text** option (for example, **log using mywork, text**), you can specify plain text by including the **.log** extension (for example, **log using mywork.log**).

Tip: Plain text logs by default. We prefer plain text for output rather than SMCL. Typing **set logtype text** at the beginning of a Stata session makes plain text the default for log files for the current session. Typing **set logtype text, permanently** makes plain text the default for the current and future sessions. In practice, we always use plain text log files; we do not find the advantages of SMCL to be enough to offset the disadvantage of only being able to read the files within Stata. If you have a question and would like to send us a log file, please send it in text format rather than in SMCL!

Closing a log file

To close a log file, type

```
. log close
```

When you exit Stata, the log file closes automatically.

Viewing a log file

Regardless of whether a log file is open or closed, a log file can be viewed in the Viewer by selecting **File→Log→View...** from the menu. When in the Viewer, you can print the log by selecting **File→Print**.

Converting from SMCL to plain text or PostScript

If you want to convert a log file from SMCL format into plain text, you can use the `translate` command. For example,

```
. translate mylog.smcl mylog.log, replace
(file mylog.log written in .log format)
```

converts the SMCL file `mylog.smcl` into the plain-text file `mylog.log`. Or, you can convert a SMCL file into a PostScript file, which is useful if you are using \TeX or \LaTeX . For example,

```
. translate mylog.smcl mylog.ps, replace
(file mylog.ps written in .ps format)
```

You can also convert a SMCL file into a PDF file:

```
. translate mylog.smcl mylog.pdf, replace
(file mylog.pdf written in PDF format)
```

Conversions can also be done through the menus by selecting **File**→**Log**→**Translate**.

2.7 Using and saving datasets

2.7.1 Data in Stata format

Stata uses its own data format with the extension `.dta`. The `use` command loads such data into memory. Pretend that we are working with the file `binlfp4.dta` in the directory `d:\spostdata`. We can load the data by typing

```
. use d:\spostdata\binlfp4, clear
```

where the `.dta` extension is assumed by Stata. The `clear` option erases all data currently in memory and proceeds with loading the new data. (Stata does not give an error if you include `clear` when there are no data in memory.) If `d:\spostdata` was our working directory, we could use the simpler command

```
. use binlfp4, clear
```

In practice, we almost always keep data in our working directory.

If you have changed the data by deleting cases, merging in another file, or creating new variables, you can save the file with the `save` command. For example,

```
. save d:\spostdata\binlfp4_V2, replace
```

where we did not need to include the `.dta` extension. We saved the file with a different name so that we can use the original data later. The `replace` option indicates that if `binlfp4_V2.dta` already exists, Stata should overwrite it. (If the file does not already exist, `replace` is ignored.) If `d:\spostdata` was our working directory, we could save the file with


```
. save binlfp4_V2, replace
```

`save` stores the data in Stata's current format, which sometimes changes with a new version of Stata, including with Stata 13. This means that a dataset saved in Stata 13 cannot be opened in Stata 12, but a dataset saved in Stata 11 or 12 can be opened in Stata 13. The `saveold` command writes the dataset in a format that is compatible with the prior format of data in Stata. If you save a dataset using `saveold` in Stata 13, you can use it in Stata 11 or 12 but not in Stata 10.

2.7.2 Data in other formats

To load data from another statistical package, such as SAS or SPSS, you need to convert it into Stata's format. The easiest way to do this is with a conversion program such as Stat/Transfer (<http://www.stattransfer.com>). We recommend obtaining one of these programs if you are using more than one statistical package or if you often share data with others who use different packages. The free statistics package R also has utilities that allow reading and writing data in different formats, including Stata format, although value labels and notes cannot be transferred this way. Stata has a set of `import` and `export` commands that allow you to deal with data that is in Microsoft Excel format, various ASCII formats, ODBC format, SAS export files, and XML. For details, type `help import` or `help export`, which include links to videos that illustrate these commands (see also [D] `import` and [D] `export`).

2.7.3 Entering data by hand

Data can also be entered by hand with a spreadsheet-style editor. Although we do not recommend using Stata's Data Editor to change existing data, because it is too easy to make a mistake, we find the Editor useful for entering small datasets. To open the Editor, type `edit` on the command line. Stata's *Getting Started* manual has a tutorial for the Data Editor, but most people who have used a spreadsheet before will be immediately comfortable. It is also easy to import data from Microsoft Excel into Stata. So, you could enter data in Excel or another spreadsheet program with which you are comfortable and which allows saving in Excel format (or `.xml`), and then you could import the data.

As you use Stata's Data Editor, every change that you make to the data is reported in the Results window and is captured by the log file (if it is open). For example, if you change `age` for the fifth observation to 32, Stata reports `replace age = 32 in 5`. This tells you that instead of using the Editor, you could have changed the data with a `replace` command. When you close the Editor, Stata asks if you want to keep the changes or revert to the unaltered data.

2.8 Size limitations on datasets

If you receive the error message `no room to add more variables, r(900) or system limit exceeded--see manual, r(1000)` when you try to load a dataset or add a variable, your dataset might have too many variables or be too large. These limits depend on the version of Stata that you are using: 32,767 variables in Stata/SE and Stata/MP, 2,047 in Stata/IC, and 99 variables in Small Stata. For details on other size limits, type the command `help limits`. If a dataset is too large for your version of Stata, you can use transfer programs such as `Stat/Transfer` to drop specified variables and optimize variable storage.

2.9 Do-files

You can execute commands in Stata by typing one command at a time into the Command window and pressing `Enter`, as we have been doing. This interactive mode is useful when you are learning Stata, exploring your data, or experimenting with alternative specifications of your regression model.

You can also create a text file containing a series of commands and then tell Stata to execute all the commands in that file, one after the other. These files, which are known as do-files because they use the extension `.do`, have the same function as syntax files in SPSS or batch files in other statistics packages. For serious work, we always use do-files because they make it easier to redo the analysis later with small modifications and because they provide an exact record of what has been done.

To get an idea of how do-files work, consider the file `example.do` saved in the working directory:

```
log using example, replace text
use binlfp4, clear
tabulate hc wc, row nlabel
log close
```

To execute a do-file, you type the command

```
do dofilename
```

in the Command window. For example, `do example` tells Stata to run each of the commands in `example.do`, one after the other. If the do-file is not in the working directory, you need to specify the directory path, such as `do d:\spostdata\example`. When Stata executes `example.do` it begins by opening the log `example.log`, then loads `binlfp4.dta`, and finally constructs a table with `hc` and `wc`. Here is what the log file looks like:

```

name: <unnamed>
log: d:\spostdata\example.log
log type: text
opened on: 01 Mar 2014, 05:04:21

. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. tabulate hc wc, row nolabel

```

Key			
<i>frequency</i>			
<i>row percentage</i>			

Husband attended college?	Wife attended college?		Total
	0	1	
0	417	41	458
	91.05	8.95	100.00
1	124	171	295
	42.03	57.97	100.00
Total	541	212	753
	71.85	28.15	100.00

```

. log close

```

2.9.1 Adding comments

Comments provide documentation within programs. Comments are simply reprinted as output; they are not treated as commands. Stata has four methods for adding comments to your do-file, and we use them all.

1. Any line that begins with `*` is a comment. Because `*` is also used as the multiplication operator for expressions and as a wildcard operator for variable names and filenames, `*` must appear at the beginning of a line to denote a comment.
2. Text following `//` is a comment, but `//` can appear anywhere within the line, not just at the beginning.
3. Text following `///` is also a comment except Stata will treat the next line as a continuation of the current line before the `///`. As we will discuss shortly, this is our preferred way of handling long command lines.
4. Everything between `/*` and `*/` is treated as a comment, no matter how many lines this spans.

The following do-file executes the same commands as the one above but includes comments:

```
/*
    ==> short simple do-file
    ==> for didactic purposes
*/

log using example, replace // this comment is ignored

// next we load the data
use binlfp4, clear

// tabulate husband's and wife's education
tabulate hc wc, /// the next line is treated as a continuation of this one
               row nlabel

// close the log file
log close
```

If you look at the do-files in the `spost13.do` package that reproduce the examples in this book, you will see that we use many comments. For serious work, we cannot emphasize enough the importance of including them. Comments are indispensable if others will be using your do-files or examining the log files, or if there is a chance that you will use them again later.

2.9.2 Long lines

Sometimes you need to execute a command that is longer than the text that fits on the screen. If you are typing the command interactively, the Command window simply pushes the left part of the command off the screen as space is needed. In general, we recommend that none of your lines in a do-file extend beyond column 80. If the lines are too long, they will either wrap in the output, which makes it harder to read, or be truncated.

You can deal with long commands in two ways. Specifying `///` at the end of a line in a do-file tells Stata to ignore the line break. Accordingly, the current line and next line(s) are be treated as one command. For example,


```
recode income91 1=500 2=1500 3=3500 4=4500 5=5500 6=6500 7=7500 8=9000    ///
          9=11250 10=13750 11=16250 12=18750 13=21250 14=23750 15=27500 16=32500 ///
          17=37500 18=45000 19=55000 20=67500 21=75000 **.
```

You can also use the `#delimit ;` command. This tells Stata to interpret `;` as the end of the command instead of interpreting a carriage return as the end of the command. (A carriage return is the character created when you press the Enter key.) After the long command is complete, you can run `#delimit cr` to return to using the carriage return as the end-of-line delimiter. For example,


```
#delimit ;
recode income91 1=500 2=1500 3=3500 4=4500 5=5500 6=6500 7=7500 8=9000
          9=11250 10=13750 11=16250 12=18750 13=21250 14=23750 15=27500 16=32500
          17=37500 18=45000 19=55000 20=67500 21=75000 **., ;
#delimit cr
```


Some other programs use `;` as the line terminator and users familiar with those programs may be more comfortable using `#delimit`. Most often we use `///` for long commands. The one time in which we use `#delimit ;` is when typing a very long `graph` command.

2.9.3 Stopping a do-file while it is running

If you are running a command or a do-file that you want to stop before it completes execution, click on  or press `Ctrl-Break`. Sometimes you will have to do this multiple times. This is because some commands ignore the breaks when making computations that should not be interrupted.

2.9.4 Creating do-files

Using Stata's Do-file Editor

Do-files can be created with Stata's built-in Do-file Editor. To use the Editor, type the command `doedit` to create a file to be named later or type `doedit filename` to create or edit a file named *filename.do*. You can also click on . The Do-file Editor is easy to use and works like most text editors except that it has several features designed specifically to support writing do-files for Stata.

- Syntax highlighting assigns colors to different keywords in Stata; the color of the text changes automatically as you type. For example, if you are typing the `poisson` command, the letters `poisso` will appear in black until you add the last `n`, at which point the entire word becomes blue. If you mistakenly type `poison`, the command remains black to indicate that you have the name of the command wrong. Syntax highlighting may sound merely decorative if you do not have much experience with coding, but it is actually extremely helpful for detecting and preventing simple errors.
- You can highlight a section of the do-file, and Stata will execute only the commands that you have highlighted. You can also execute only the commands starting wherever the cursor is located to the end of the file.
- A gray vertical line in the Editor indicates where column 80 is, to nudge you to try to keep individual lines shorter than this.
- Auto-indenting and line folding are features that will become very useful as you write more complicated programs that use loops.

If you have not used the Do-file Editor since earlier versions of Stata, we encourage you to try it again with the many new features added in recent years. Spending some time up-front reading [GS] 13 **Using the Do-file Editor** can save you a lot of time in the long run.

Using other editors to create do-files

Because do-files are plain text files, you can create do-files with any program that creates text files. Specialized text editors work much better than word processors such as Microsoft Word. We put this emphatically: If you are writing do-files using a word processor, you are making life far more difficult than it needs to be. Among other things, with word processors it is easy to forget to save the file as plain text. While one of the authors prefers to use Stata's built-in Do-file Editor, the other prefers a stand-alone editor that has many features, making it faster to create do-files. For example, you can create templates that quickly insert commonly used commands.

If you use an editor other than Stata's built-in Do-file Editor, you might not be able to run the do-file by clicking on an icon or selecting from a menu.¹ Instead, you will need to switch from your editor and then type the command `do filename`.

Warning. Stata executes commands when it encounters a line break (also called a carriage return, created when you press the **Enter** key). If you do not include a line break after the last line in a do-file, that last line will not be executed. Stata's Do-file Editor handles this automatically by default.

2.9.5 Recommended structure for do-files

This is the basic structure that we recommend for do-files:

```
1] capture log close
2] log using <filename>, replace text
3] version 13.1
4] set linesize 80
5] set scheme s2color
6] clear all
7] macro drop _all

8] // task:
9] // #1
10] // #2

11] log close
12] exit
```

1. Friedrich Huebler's blog (2013) has information on how to integrate Stata with some external text editors. We have not, however, tried this.

Because we hope you will use do-files a lot, let's briefly review these commands.

Lines 1–2. The command `capture log close` is very useful. Suppose you have a do-file that starts with `log using mylog, replace`. You run the file and it “crashes” before reaching `log close`, which means that the log file remains open. If you revise the do-file and run it again, an error is generated when Stata tries to open the log file because the file is already open. The prefix `capture` tells Stata not to stop the do-file if the command that follows produces an error (see [P] `capture`). Accordingly, `capture log close` closes the log file if it is open. If it is not open, the error generated by trying to close an already-closed file is ignored. Because of line 1, line 2 will never generate the error that the log is already open.

Line 3. The `version 13.1` command indicates that the program was written for use in Stata 13.1. This command tells any future version of Stata that you want the commands that follow to work just as they did in Stata 13. This prevents the problem of old do-files not running correctly in newer releases of the program. If you place the `version` command after the `log` command, you can confirm the version that was used to generate the output in the log.

Lines 4–5. By setting the line size within the do-file, you can be sure that the output will look the same if you later run the do-file with the line size set to some other value. The `set scheme` command controls how graphs will look. By including this line in the do-file, graphs should look the same when you run the do-file later.

Lines 6–7. These lines remove everything from memory. Although it seems that `clear all` would do this by itself, `clear all` does not remove macros. Accordingly, we drop them in line 7. We prefer a do-file to be self contained, meaning that it does not depend on anything in memory from interactive commands or other do-files that were run in the same session. This way, the do-file will run the same way later.

Lines 8–10. The comment `// task:` is where we explain what the do-file is doing. This might require multiple lines. We generally like to include what is being done, what project it is associated with, who did it, and when. This information proves very useful when you return to the do-file later. For long do-files, we find it is also useful to add numbered comments that explain what each major step of the program is doing. This is particularly handy when you are working collaboratively and discussing the output over the phone or by email.

Lines 11–12. Line 11 closes the log file, but this command will only run if line 11 ends with a carriage return (obtained when you press the Enter key on your keyboard). Including `exit` in line 12 is not required, but it is helpful in two ways: First, you know that line 11 will execute, because it is followed by another line, which required a carriage return at the end of line 11. Second, because `exit` tells Stata to exit the do-file, you can use the end of your do-file for recording notes.

2.10 Using Stata for serious data analysis

Voltaire is said to have written *Candide* in three days. Creative work often rewards such inspired, seat-of-the-pants, get-the-details-later activity—data management does not. Instead, effective data management rewards forethought, carefulness, double- and triple-checking of details, and meticulous, albeit tedious, documentation. Errors in data management are astonishingly (and painfully) easy to make. Moreover, tiny errors can have disastrous implications that can cost hours and even days of work. The extra time it takes to conduct data management carefully is rewarded many times over by the reduced risk of errors. That is, it helps prevent you from getting incorrect results that you do not know are incorrect.

With this in mind, we begin with some broad, perhaps irritatingly practical, suggestions for doing data analysis efficiently and effectively.

1. *Ensure replicability by using do-files and log files for everything.* For data analysis to be credible, you must be able to reproduce entirely and exactly the trail from the original data to the tables and graphs in your paper. Thus any permanent changes you make to the data should be made by running do-files rather than by using the interactive mode. If you work interactively, be sure that the first thing you do is to open a log file. Then when you are done, you can use these files to create a do-file to reproduce your interactive results. Stata's `datasignature` command (see [D] `datasignature`) also provides a way to ensure that the values in a dataset you are using are exactly the same as those used earlier.
2. *Document your do-files.* Reasoning that is obvious today can be baffling in six months. We use comments extensively in our do-files—they are invaluable for remembering what we did and why we did it (or for sharing our code with others). If you use intuitive names for variables and for files, that will also make it easier to figure out or remember what a do-file is doing.
3. *Keep a research diary.* For serious work, you should keep a diary that includes a description of every program you run, the research decisions that are being made (for example, the reasons for recoding a variable in a particular way), and the files that are created. A good research diary allows you to reproduce everything you have done starting with the original data. We cannot emphasize enough how helpful such notes are when you return to a project that was put on hold, when you are responding to reviewers, or when you are moving on to the next stage of your research.
4. *Develop a system for naming files.* Usually, it makes the most sense to have each do-file generate one log file with the same prefix (for example, `clean_data.do`, `clean_data.log`). Names are easiest to organize when brief, but they should be long enough and logically related enough to make sense of the task the file does. One author prefers short names, organized by major task (for example, `recode01.do`), whereas the other author likes longer names (for example, `MakeIncomeVars.do`). Use whatever works best for you.

5. *Use new names for new variables and files.* Never change a dataset and save it with the original name. If you drop three variables from `pcoms1.dta` and create two new variables, call the new file `pcoms2.dta`. When you transform a variable, give it a new name rather than simply replacing or recoding the old variable. For example, if you have a variable called `workmom` with a five-point attitude scale, and you want to create a binary variable indicating positive and negative attitudes, create a new variable called `workmom2`.
6. *Use labels and notes.* When you create a new variable, give it a variable label. If it is a categorical variable, assign value labels. You can add a note about the new variable by using the `notes` command (described below). When you create a new dataset, you can also use `notes` to document what it is.
7. *Double-check every new variable.* Cross-tabulating or graphing the old variable and the new variable are often effective strategies for verifying new variables. As we describe below, using `list` with a subset of cases is similarly effective for checking transformations. Be sure to look carefully at the frequency distributions and summary statistics of variables in your analysis. You would not believe how many times puzzling regression results turn out to involve miscodings of variables that would have been immediately apparent by looking at the descriptive statistics.
8. *Practice good archiving.* If you want to retain hard copies of all your analyses, develop a system of binders for doing so rather than a set of intermingling piles on your desk. Otherwise, maintain an orderly set of directories and filenames rather than creating files haphazardly. Back up everything, preferably with a system that works automatically and saves older versions of files as well. Make off-site or Cloud-based backups or keep any on-site backups in a fireproof box. Should cataclysm strike, you will have enough other things to worry about without also having lost months or years of work.

Long's (2009) *The Workflow of Data Analysis Using Stata* considers all of these issues in detail. Long presents methods for planning, organizing, and documenting your research to make your work efficient but also, most importantly, to increase the reliability and replicability of your research.

2.11 Syntax of Stata commands

Think about the syntax of commands in everyday, spoken English. They usually begin with a verb telling the other person what to do. Sometimes the verb is the entire command: "Help!" or "Stop!" Sometimes the verb needs to be followed by an object that indicates whom or what the verb is to be performed on: "Help Dave!" or "Stop the car!" Sometimes the verb is followed by a qualifier that gives specific conditions under which the command should or should not be performed: "Give me a piece of pizza *if it does not have mushrooms*" or "Call me *in 10 minutes*". Verbs can also be followed by adverbs that specify a particular way the action should be performed, such as when a teacher commands her students to "Talk *clearly*" or "Walk *single file*".

Stata follows an analogous logic, albeit with some wrinkles that we will introduce later. The basic syntax of a command has four parts:

1. *Command*: What action do you want performed?
2. *Names of variables, files, or other objects*: On what things is the command to be performed?
3. *Qualifier on observations*: On which observations should the command be performed?
4. *Options*: What special things should be done when executing the command?

All commands in Stata require the first of these parts, just as a spoken command requires a verb. Each of the other three parts can be required, optional, or not allowed, depending on the particular command and circumstances. To illustrate how commands work before we get into the details, we use the `tabulate` command to make a two-way table of the frequencies of variables `hc` by `wc`:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. tabulate hc wc if age>40, row
```

Key			
frequency			
row percentage			
Husband attended college?	Wife attended college?		Total
	no	college	
no	263 91.96	23 8.04	286 100.00
college	58 38.93	91 61.07	149 100.00
Total	321 73.79	114 26.21	435 100.00

By putting `hc` before `wc`, we make `hc` the row variable and `wc` the column variable. The condition `if age>40` specifies that the frequencies should include observations only for those older than 40. The option `row` indicates that row percentages should be printed as well as frequencies. These percentages allow us to see that in 61% of the cases in which the husband had attended college, the wife had also done so, whereas wives had attended college in only 8% of cases in which the husbands had not. Notice the comma preceding `row`: whenever options are specified, they are at the end of the command with a single comma to indicate the beginning of the list of options. The precise ordering of multiple options after the comma is never important.

Next, we provide more information on each of the four components of syntax.

2.11.1 Commands

Commands define the tasks that Stata is to perform. A great thing about Stata is that the set of commands is completely open ended. It expands not just with new releases of Stata but also when users add their own commands, such as our added `SPost` commands. Each new command is stored in its own file, ending with the extension `.ado`. Whenever Stata encounters a command that is not in its built-in library, it searches various directories for the appropriate `ado`-file. A list of the directories it searches and the order in which it searches them can be obtained by typing `adopath`. This list includes all the places Stata intends for storage of official and user-written commands.

2.11.2 Variable lists

Variable names are case sensitive. For example, you could have three different variables named `income`, `Income`, and `inCome`. Of course, this is not a good idea because it leads to confusion. To keep life simple, we typically stick to lowercase names. Stata allows variable names up to 32 characters long, compared with the 8 character maximum imposed by earlier versions of Stata and by some other statistics packages. We recommend using shorter names because longer variable names become unwieldy to type. We also recommend using variable names that have some mnemonic value because completely arbitrary names (for example, `var00031`) invariably lead to confusion. Although variable names can be abbreviated to the initial set of characters that identifies the variable uniquely, we worry that too much reliance on this feature might cause mistakes to be made.

Many commands assume that if you have not listed any variables as arguments, then you want to perform the operation on every variable in the dataset. For example, the `summarize` command provides summary statistics on the listed variables:

```
. summarize inc k5 wc
```

Variable	Obs	Mean	Std. Dev.	Min	Max
inc	753	20.12897	11.6348	-.0290001	96
k5	753	.2377158	.523959	0	3
wc	753	.2815405	.4500494	0	1

We could also get summary statistics on every variable in our dataset by just typing `summarize` without any variables listed:

```
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
caseid	753	377	217.5167	1	753
lfp	753	.5683931	.4956295	0	1
k5	753	.2377158	.523959	0	3
k618	753	1.353254	1.319874	0	8
age	753	42.53785	8.072574	30	60
<hr/>					
wc	753	.2815405	.4500494	0	1
hc	753	.3917663	.4884694	0	1
lwg	753	1.097115	.5875564	-2.054124	3.218876
inc	753	20.12897	11.6348	-.0290001	96
age3039	753	.3957503	.4893363	0	1
<hr/>					
age4049	753	.3851262	.4869486	0	1
age50plus	753	.2191235	.4139274	0	1
agecat	753	1.823373	.7644952	1	3
k5_0	753	.8047809	.3966327	0	1
k5_1	753	.1567065	.3637655	0	1
<hr/>					
k5_2	753	.0345286	.1827038	0	1
k5_2plus	753	.0385126	.1925581	0	1
k5_3	753	.0039841	.0630354	0	1
k5cat	753	.2337317	.5064257	0	2
k618_0	753	.3426295	.4749042	0	1
<hr/>					
k618_1	753	.2456839	.4307781	0	1
k618_23	753	.3519256	.4778883	0	1
k618_4plus	753	.059761	.237201	0	1
k618cat	753	1.128818	.9582402	0	3
wages	753	3.566993	2.644393	.1282051	25

You can also select all variables that begin or end with the same letters by using the wildcard operator `*`. For example, to `summarize` all variables that begin with `l`, type

```
. summarize l*
```

Variable	Obs	Mean	Std. Dev.	Min	Max
lfp	753	.5683931	.4956295	0	1
lwg	753	1.097115	.5875564	-2.054124	3.218876

Tip: Removing the separator. If you do not like the horizontal separator that appears after every five variables in the output for `summarize`, you can remove it with the option `sep(0)`, such as `summarize, sep(0)`.

2.11.3 if and in qualifiers

Stata has two qualifiers that restrict the sample being analyzed: **if** and **in**. The **in** qualifier performs operations on a range of consecutive observations. Typing `summarize in 20/100` gives summary statistics for only the 20th through the 100th observations. **in** restrictions depend on the current sort order of the data, meaning that if you re-sort your data, the 81 observations selected by the restriction `summarize in 20/100` might be different.²

In practice, **if** conditions are used much more often than **in** conditions. The **if** qualifier restricts the observations to those that fulfill a specified condition. For example, `summarize if age<50` provides summary statistics for only those observations where **age** is less than 50. Here is a list of operators that can be used to construct logical **if** statements:

Operator	Definition	Example
<code>==</code>	Equal to	<code>if female==1</code>
<code>!=</code>	Not equal to	<code>if female!=1</code>
<code>></code>	Greater than	<code>if age>20</code>
<code>>=</code>	Greater than or equal to	<code>if age>=21</code>
<code><</code>	Less than	<code>if age<66</code>
<code><=</code>	Less than or equal to	<code>if age<=65</code>
<code>&</code>	And	<code>if age==21 & female==1</code>
<code> </code>	Or	<code>if age==21 educ>16</code>

Two notes about the **if** qualifier:

1. Use a double equal sign (for example, `summarize if female==1`) to specify a condition to test. When assigning a value to something, such as when creating a new variable, use a single equal sign (for example, `gen newvar = 1`). Putting these examples together results in `gen newvar = 1 if female==1`.
2. A missing-value code is treated as the largest positive number when evaluated with an **if** condition. In other words, Stata treats missing cases as positive infinity when evaluating **if** expressions. If you type `summarize ed if age>50`, the summary statistics for **ed** are calculated on all observations where **age** is greater than 50, including cases where the value of **age** is missing. You must be careful of this when using **if** with `>` or `>=` operators. If you type `summarize ed if age<.`, Stata gives summary statistics for cases where **age** is not missing. Entering `summarize ed if age>50 & age<.` provides summary statistics for those

2. No Stata command should change the sort order of the data—unless that is the purpose of the command—but readers should beware that user-written programs may not always follow proper Stata programming practice.

cases where **age** is greater than 50 and is not missing. See section 2.12.3 for more details on missing values.

Examples of if qualifier

If we wanted summary statistics on income for only those respondents who were between the ages of 25 and 65, we would type

```
. summarize income if age>=25 & age<=65
```

If we wanted summary statistics on income for only female respondents who were between the ages of 25 and 65, we would type

```
. summarize income if age>=25 & age<=65 & female==1
```

If we wanted summary statistics on income for the remaining female respondents—that is, those who are younger than 25 or older than 65—we would type

```
. summarize income if (age<25 | age>65) & age<. & female==1
```



We need to include **& age<.** because otherwise, the condition **(age<25 | age>65)** would include those cases for which **age** is missing.

2.11.4 Options

Options are set off from the rest of the command by a comma. Options can often be abbreviated, although whether and how they can be abbreviated varies across commands. In this book, we rarely cover all the options available for any given command, but you can use **help** to see them all.

2.12 Managing data

2.12.1 Looking at your data

Two easy ways to look at your data are **browse** and **list**. The **browse** command opens a spreadsheet (in the Browser) in which you can scroll to view the data but you cannot change the data. You can view and change data with the **edit** command (in the Data Editor), but this is risky. We much prefer making changes to our data using do-files, even when we are changing the value of only one variable for one observation. The Browser is also available by clicking on , and the Data Editor is also available by clicking on .

The **list** command creates a list of values of specified variables and observations. **if** and **in** qualifiers can be used to look at just a portion of the data, which is sometimes useful for checking that transformations of variables are correct. For example, if you want to confirm that the variable **lninc** has been correctly constructed as the natural

log of `inc`, you could type `list inc lninc in 1/20` to the values of `inc` and `lninc` for the first 20 observations.

2.12.2 Getting information about variables

You can obtain basic information about your variables in several ways. Here are the commands that we find useful. Which one you use will depend mostly on the kind and level of detail you need.

`codebook`, `compact`. This command gives you basic descriptive statistics along with variable labels:

```
. codebook lfp k5 k618 agecat wc hc lwg inc, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
lfp	753	2	.5683931	0	1	In paid labor force?
k5	753	4	.2377158	0	3	# kids < 6
k618	753	9	1.353254	0	8	# kids 6-18
agecat	753	3	1.823373	1	3	Wife's age group
wc	753	2	.2815405	0	1	Wife attended college?
hc	753	2	.3917663	0	1	Husband attended college?
lwg	753	676	1.097115	-2.054124	3.218876	Log of wife's estimated...
inc	753	621	20.12897	-.0290001	96	Family income excluding...

We often use this command at the start of do-files to provide a summary of the variables being analyzed. While `codebook`, `compact` does include the variable label, it does not include the standard deviation.

`summarize`. To get descriptive statistics including the standard deviation, we use the `summarize` command, which does not include the variable label. By default, `summarize` presents the number of nonmissing observations, the mean, the standard deviation, the minimum values, and the maximum values. Adding the `detail` option includes more information. For example,

```
. summarize inc, detail
```

Family income excluding wife's				
Percentiles		Smallest		
1%	3.777	-.0290001		
5%	7.044	1.2		
10%	9.02	1.5	Obs	753
25%	13.025	2.134	Sum of Wgt.	753
50%	17.7		Mean	20.12897
		Largest	Std. Dev.	11.6348
75%	24.466	79.8		
90%	32.7	88	Variance	135.3685
95%	41.1	91	Skewness	2.210531
99%	68.035	96	Kurtosis	11.38358

`tabulate` and `tab1`. The `tabulate` command creates the frequency distribution for a variable. For example,

```
. tabulate hc
```

Husband attended college?	Freq.	Percent	Cum.
no	458	60.82	60.82
college	295	39.18	100.00
Total	753	100.00	

If you do not want the value labels included, add the `nolabel` option:

```
. tabulate hc, nolabel
```

Husband attended college?	Freq.	Percent	Cum.
0	458	60.82	60.82
1	295	39.18	100.00
Total	753	100.00	

If you want a two-way table, type

```
. tabulate hc wc
```

Husband attended college?	Wife attended college?		Total
	no	college	
no	417	41	458
college	124	171	295
Total	541	212	753

By default, `tabulate` does not tell you the number of missing values for either variable. You can specify the `missing` option to include missing values. We recommend this option whenever you are generating a frequency distribution to check that some transformation was done correctly. The options `row`, `col`, and `cell` request row, column, and cell percentages, respectively, along with the frequency counts. The option `chi2` reports the χ^2 for a test that the rows and columns are independent.

Although you need a separate `tabulate` command for each variable, `tab1` computes univariate frequency distributions for each variable listed. For example,

```
. tab1 hc wc
```

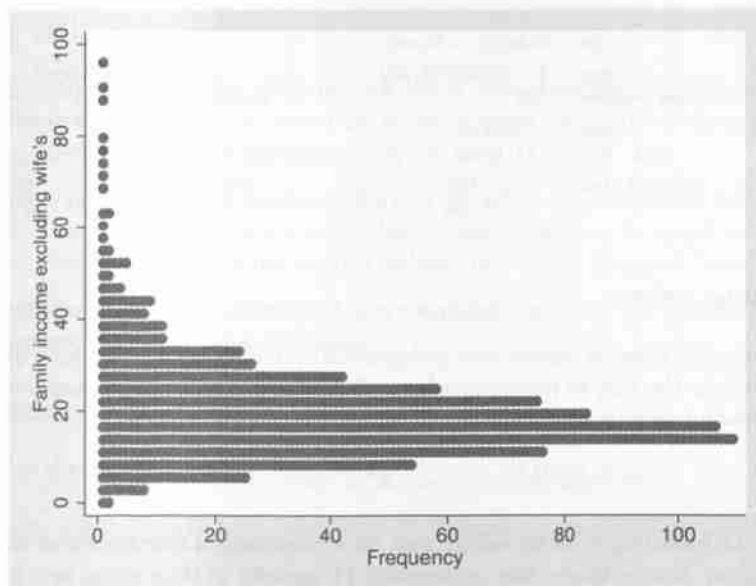
```
-> tabulation of hc
```

Husband attended college?	Freq.	Percent	Cum.
no	458	60.82	60.82
college	295	39.18	100.00
Total	753	100.00	

```
-> tabulation of wc
```

Wife attended college?	Freq.	Percent	Cum.
no	541	71.85	71.85
college	212	28.15	100.00
Total	753	100.00	

`dotplot`. This command provides a quick graphical summary of a variable, which is useful for checking your data. For example, `dotplot inc` leads to the following graph:



Details on saving, printing, and enhancing graphs are given in section 2.17.

codebook and describe. If you click on a variable in the Variables window, information on that variable will be displayed in the Properties window. You will see the name of the variable, the labels associated with it, and its storage and display formats. If you have saved notes about the variable with the `notes` command (see 2.14.3 below and [D] `notes`), these are also displayed in this window. The same information (except for the notes) is available using the `describe` command, and `describe` can be used to list this information for multiple variables at once. For example,

```
. describe lfp k5 k618 agecat wc hc lwg inc
```

variable name	storage type	display format	value label	variable label
lfp	byte	%9.0g	lfp	In paid labor force?
k5	byte	%9.0g		# kids < 6
k618	byte	%9.0g		# kids 6-18
agecat	byte	%9.0g	agecat	* Wife's age group
wc	byte	%9.0g	Lcol	Wife attended college?
hc	byte	%9.0g	Lcol	Husband attended college?
lwg	float	%9.0g		Log of wife's estimated wages
inc	float	%9.0g		Family income excluding wife's

The `codebook` command also provides more detailed information in a format designed for printing a codebook. For example,

```
. codebook inc
```

inc		Family income excluding wife's				
type:	numeric (float)					
range:	[-.02900009,96]	units:	1.000e-09			
unique values:	621	missing :	0/753			
mean:	20.129					
std. dev:	11.6348					
percentiles:	10%	25%	50%	75%	90%	
	9.02	13.025	17.7	24.466	32.7	

2.12.3 Missing values

Although numeric missing values are automatically excluded when Stata fits models, they are stored as the largest positive values. Twenty-seven missing values are available, with the following ordering:

all numbers < . < .a < .b < ... < .z

This way of handling missing values can have unexpected consequences when determining samples. For instance, the expression `if age>65` is true when `age` has a value greater than 65 and when `age` is missing. Similarly, the expression `if occupation!=1` is true if `occupation` is not equal to 1 or if `occupation` is missing. When expressions such as these are required, be sure to explicitly exclude any unwanted missing values.

For instance, if `age>65 & age<.` would be true only for those people whose age is not missing and who are over 65. Similarly, if `occupation!=1 & occupation<.` would be true only when the `occupation` is not missing and is not equal to 1.

The different missing values can be used to record the distinct reasons why a variable is missing. For instance, consider a survey that asked people about their driving records and contains a variable to record whether the respondent received a ticket after being involved in an accident. Any missing variables could be denoted `.a` to indicate the respondent had not been involved in any accidents or denoted `.b` to indicate the respondent refused to answer the question.

2.12.4 Selecting observations

As previously mentioned, you can select specific sets of observations with the `if` and `in` qualifiers; for example, `summarize age if wc==1` provides summary statistics on `age` for only those observations where `wc` equals 1. Sometimes it is simpler to remove the observations with either the `drop` or the `keep` command. These commands remove or keep observations from memory (not from the `.dta` file) based on an `if` or `in` specification. The syntax is

```
drop [in] [if]
```

or

```
keep [in] [if]
```

With `drop`, only observations that do not meet the specified conditions are left in memory. For example, `drop if wc==1` keeps only those cases where `wc` is not equal to 1, including observations with missing values on `wc`.

With `keep`, only observations that meet the specified conditions are left in memory. For example, `keep if wc==1` keeps only those cases where `wc` is equal to 1; all other observations, including those with missing values for `wc`, are dropped from memory.

After selecting the observations that you want, you can save the remaining variables to a new dataset with the `save` command.

2.12.5 Selecting variables

You can also simply select which variables you want to drop or keep. The syntax is

```
drop variable_list
```

or

```
keep variable_list
```


With **drop**, all variables are kept except those that are explicitly listed, which are dropped. With **keep**, only those variables that are explicitly listed are kept. After selecting the variables that you want, you can save the remaining variables to a new dataset with the **save** command.

2.13 Creating new variables

The variables that you analyze are often constructed differently from the variables in the original dataset. Here we consider basic methods for creating new variables. Our examples always create a new variable from an old variable rather than transforming an existing variable. Even though you can simply transform an existing variable, we find that this can lead to mistakes.

In addition to the commands discussed in this section, you can use Stata's factor-variable notation, which tells Stata how to create new variables "on the fly" from existing variables during the analysis. For example, specifying **regress y c.age c.age#c.age** tells Stata to run a regression with the independent variable **age** and **age**-squared. Only the variable **age** is in the dataset, but **c.age#c.age** tells Stata that **age** is a continuous variable and that a term for the square of **age** should be added to the regression. Factor-variable notation allows you to create powers of variables, interactions between variables, and indicator variables. This valuable tool, which is used throughout the book, is discussed in detail in section 3.1.5.

2.13.1 The generate command

The **generate** command creates new variables. For example, to create **age2** as an exact copy of **age**, type

```
. generate age2 = age
. summarize age2 age
```

Variable	Obs	Mean	Std. Dev.	Min	Max
age2	753	42.53785	8.072574	30	60
age	753	42.53785	8.072574	30	60

The results of **summarize** show that the two variables are identical. We used a single equal sign because we are making a variable equal to some value.

Observations excluded by **if** or **in** qualifiers in the **generate** command are coded as missing. For example, to generate **age3** that equals **age** for those over 40 but is otherwise missing, type


```
. gen age3 = age if age>40
(318 missing values generated)
. summarize age3 age
```

Variable	Obs	Mean	Std. Dev.	Min	Max
age3	435	48.3977	4.936509	41	60
age	753	42.53785	8.072574	30	60

Whenever **generate** (or **gen**, as it can be abbreviated) produces missing values, it tells you how many cases are missing.

generate can also create variables that are mathematical functions of existing variables. For example, we can create **agesq** that is the square of **age** and create **lnage** that is the natural log of **age**:

```
. gen agesq = age^2
. gen lnage = ln(age)
```

For quick reference, here is a list of the standard mathematical operators:

Operator	Definition	Example
+	Add	gen y = a+b
-	Subtract	gen y = a-b
/	Divide	gen density = pop/area
*	Multiply	gen y = a*b
^	Take to a power	gen y = a^3

Here are some particularly useful functions:

Function	Definition	Example
ln()	Natural log	gen lnwage = ln(wage)
exp()	Exponential	gen y = exp(a)
sqrt()	Square root	gen agesqrt = sqrt(age)

For a complete list of functions, type **help functions**. The functions we list above are listed under *Mathematical functions*. For working with probability distributions, the functions in the *Probability distributions and density functions* category can be very helpful. For working with string (text) variables, the functions in the *String functions* category are valuable.

Tip: Using clonevar instead of generate. Although `generate newvar = oldvar` is the most intuitive way of creating a copy of the values of *oldvar* as *newvar*, sometimes `clonevar newvar = oldvar` is a better alternative. `clonevar` copies not only the values of *oldvar* but also the variable label, value label, and other attributes. Note that if you usually give value labels the same name as the variables they apply to, `clonevar` does not re-create or rename value labels. Instead, the label names for *oldvar* also will be attached to *newvar*.

Using the `generate` command with various functions is only one way to create new variables. A second approach is to use the `egen` command, which includes some powerful tools for creating variables. For example, the `egen std()` function standardizes variables by subtracting the mean and dividing by the standard deviation. The `egen rowmean()` function will compute a new variable that is the mean of a set of variables (although be sure to check the help file for how missing-variable values are handled). Many `egen` functions also support the `by varlist:` prefix that allows you to compute functions based on group-specific values. For example, `by female: egen std(varname)` standardizes a variable within sex, so the mean for both men and women is 0. The best way to familiarize yourself with the functionality available through `egen` is to read `help egen` or [D] `egen`.

2.13.2 The replace command

`replace` has the same syntax as `generate` but is used to change values of a variable that already exists. For example, say we want to make a new variable, `age4`, that equals `age` if `age` is over 40 but equals 40 for all persons aged 40 and under. First, we create `age4` equal to `age`. Then we replace those values we want to change:

```
. gen age4 = age
. replace age4 = 40 if age<40
(298 real changes made)
. summarize age4 age
```

Variable	Obs	Mean	Std. Dev.	Min	Max
age4	753	44.85126	5.593896	40	60
age	753	42.53785	8.072574	30	60

`replace` reports how many values were changed. This is useful in verifying that the command did what you intended. `summarize` confirms that the minimum value of `age` is 30 and that `age4` now has a minimum of 40 as intended.

Warning. We could have simply changed the original variable: `replace age = 40 if age<40`. But if we did this and saved the data, there would be no way to return to the original values for `age` if we later needed them. As a general rule, never change an existing variable. Instead, create a copy and make changes to the copy.

2.13.3 The recode command

The values of existing variables can also be changed using the `recode` command. With `recode`, you specify a set of correspondences between old values and new ones. For example, you might want old values of 1 and 2 to correspond to new values of 1, old values of 3 and 4 to correspond to new values of 2, and so on. This is particularly useful for combining categories.

Like before, we recommend that you start by making a copy of the existing variable and then recode the copy. Or, to be more efficient, you can use the `recode` command with the `generate(newvarname)` option. With this option, Stata creates a new variable instead of overwriting the old one. We include several examples of `recode` below; for more, type `help recode`.

To change 1 to 2 and 3 to 4 but leave all other values unchanged, type

```
. recode origvar (1=2) (3=4), generate(myvar1)
(23 differences between origvar and myvar1)
```

To change 2 to 1 and change all other values, including missing, to 0, type

```
. recode origvar (2=1) (*=0), generate(myvar2)
(100 differences between origvar and myvar2)
```

The asterisk (*) indicates all values, including missing values, that have not been explicitly recoded.

To change 2 to 1 and change all other values except missing to 0, type

```
. recode origvar (2=1) (nonmissing=0), generate(myvar3)
(89 differences between origvar and myvar3)
```

To change values from 1 to 4, inclusive, to 2 and keep other values unchanged, type

```
. recode origvar (1/4=2), generate(myvar4)
(40 differences between origvar and myvar4)
```

To change values 1, 3, 4, and 5 to 7 and keep other values unchanged, type

```
. recode origvar (1 3 4 5=7), generate(myvar5)
(55 differences between origvar and myvar5)
```


To change all values from the minimum through 5 to the minimum, type

```
. recode origvar (min/5=min), generate(myvar6)
(56 differences between origvar and myvar6)
```

To change missing values to 9, type

```
. recode origvar (missing=9), generate(myvar7)
(11 differences between origvar and myvar7)
```

`recode` can be used to recode several variables at once if they are all to be recoded the same way. Just include all the variable names before the instructions on how they are to be recoded and, within the parentheses of the `generate()` option, include all the names for new variables if you do not want the old variables to be overwritten.

2.14 Labeling variables and values

Variable labels provide descriptive information about what a variable measures. For example, the variable `agesq` might be given the variable label “age-squared”, or `warm` could have the label “Mother has a warm relationship”. Value labels provide labels for the different values of a categorical variable. For example, value labels might indicate that the values 1–4 correspond to survey responses of strongly agree, agree, disagree, and strongly disagree. Adding labels to variables and values is not much fun, but in the long run, it can save much time and prevent misunderstandings. We believe all variables should have variable labels, and value labels should be used for at least all ordinal and nominal variables, as well as any variables that use multiple missing value codes. Also, some of the commands in *SPost* produce output that is more easily understood if the dependent variable has value labels.

2.14.1 Variable labels

The `label variable` command attaches a label of up to 80 characters long to a variable. For example,

```
. use gsskidvalue4, clear
(1993 and 1994 General Social Survey)
. gen agesq = age*age
. label variable agesq "age-squared of respondent"
. codebook age agesq, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
age	4598	73	46.12375	18	99	age of respondent
agesq	4598	73	2427.72	324	9801	age-squared of respondent

If no label is specified, any existing variable label is removed. For example,

```
. label variable agesq
. codebook agesq, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
agesq	4598	73	2427.72	324	9801	

Tip: Use short labels. Although variable labels can be up to 80 characters long, we recommend that you strive to be concise. Output often does not show all 80 characters and truncates the label instead. For the same reason, we also find it useful to put the most important information at the beginning of the label. That way, if the label is truncated, you will still see the critical information.

Tip: Searching variable labels. Typing `lookfor string` will search the dataset in memory and list all variables in which *string* appears in either the variable name or the variable label.

2.14.2 Value labels

Beginners often find value labels in Stata confusing. What may be most nonintuitive is that Stata splits the process of labeling values into two steps: creating labels and then attaching the labels to variables.

Step 1 defines a set of labels without reference to a specific variable. Here are some examples of value labels:

```
. label define Lyesno 1 yes 0 no
. label define Lposneg4 1 veryN 2 negative 3 positive 4 veryP
. label define Lagree4 1 StrongA 2 Agree 3 Disagree 4 StrongD
. label define Lagree5 1 StrongA 2 Agree 3 Neutral 4 Disagree 5 StrongD
```

Several features of the labels demonstrated above are worth noting:

1. Each set of value labels is given a unique name (for example, `Lyesno`, `Lagree4`). We often use the convention of starting a label name with `L` if it is used with multiple variables. If a label will be used with only one variable, we give the label the name of the variable, as illustrated below. This way, if we change a label that starts with `L`, we know to make sure that the change is appropriate for all the variables that use that label.
2. Periods, colons, and curly brackets in value labels produce errors in some commands and should be avoided. Although we commonly use spaces in labels (for example, `Strong A`), in rare instances these can also cause problems.

3. Our labels are generally 10 characters or shorter because some programs have trouble with long value labels. We might use longer labels if we know the commands we are using can handle them.

Step 2 assigns the value label definitions to variables. Let's say that variables `female`, `black`, and `anykids` all imply "yes and no" categories with 1 as yes and 0 as no. To assign labels to the values, we would use the following commands:

```
. label values female Lyesno
. label values black Lyesno
. label values anykids Lyesno
. describe female black anykids
```

variable name	storage type	display format	value label	variable label
female	byte	%9.0g	Lyesno	Female
black	byte	%9.0g	Lyesno	Black
anykids	byte	%9.0g	Lyesno	R have any children?

The output for `describe` shows which value labels were assigned to which variables. The new value labels are reflected in the output from `tabulate`:

```
. tabulate anykids
```

R have any children?	Freq.	Percent	Cum.
no	1,267	27.64	27.64
yes	3,317	72.36	100.00
Total	4,584	100.00	

For the `degree` variable, we assign labels that will be used only with that variable. Accordingly, the name of the variable and the label are the same:

```
. label define degree 0 "no_hs" 1 "hs" 2 "jun_col" 3 "bachelor" 4 "graduate"
. label values degree degree
. tabulate degree
```

rs highest degree	Freq.	Percent	Cum.
no_hs	801	17.47	17.47
hs	2,426	52.92	70.40
jun_col	273	5.96	76.35
bachelor	750	16.36	92.71
graduate	334	7.29	100.00
Total	4,584	100.00	

If you want a list of the value labels being used in your current dataset, use the command `labelbook`, which provides a detailed list of all value labels, including which labels are assigned to which variables. This can be useful both in setting up a complex dataset and for documenting your data.

2.14.3 The notes command

The `notes` command allows you to add notes to the dataset as a whole or to specific variables. This is often referred to as adding metadata. Because the notes are saved in the dataset, the information is always available when you use the data. Here we add one note describing the dataset and two describing the income variable:

```
. notes: General Social Survey extract for Stata book | J Freese | 2014-01-23
. notes income: self-reported family income, measured in dollars
. notes income: refusals coded as missing
```

These notes can be viewed in the Properties window. We can also review the notes by typing

```
. notes
_dta:
  1. General Social Survey extract for Stata book | J Freese | 2014-01-23
income:
  1. self-reported family income, measured in dollars
  2. refusals coded as missing
```

If we save the dataset after adding notes, the notes become a permanent part of the dataset.

2.15 Global and local macros

Good programming and data analysis practice involves repeating oneself as little as possible. Among the most powerful and flexible tools toward this end in Stata are macros. Because later in the book we use macros extensively, we introduce them briefly here. Readers who have less familiarity with Stata might want to skim this section and the next for now and read them later when macros and loops are used in later chapters.

The term “macro” can be confusing for people familiar with its use in some other software packages, notably, Microsoft Word. We have found it is easier for people to get the hang of macros in Stata if they simply set aside any sense of the term they have acquired using other software. In Stata, a macro is a name associated with a string of characters or a number. Once a macro is created, whenever Stata encounters the macro name, it automatically substitutes the contents of the macro.

To give an example, pretend that you want to generate a series of two-by-two tables where you want cell percentages, requiring the `cell` option; missing values, requiring the `missing` option; values printed instead of value labels, requiring the `nolabel` option; the table to be printed without a key, requiring the `nokey` option; and the chi-squared test statistic, requiring the `chi2` option. Even if you use the shortest abbreviations, this would require typing `cell miss nolabel chi2 nokey` at the end of each `tabulate` command. Instead, you could use the following command to define a global macro called `myoptions`:


```
. global myoptions ", cell miss nolabel chi2 nokey"
```

Whenever you type `$myoptions` (the `$` tells Stata that `myoptions` is a global macro), Stata substitutes `, cell miss nolabel chi2 nokey`; that is, if you type

```
. tabulate lfp wc $myoptions
```

Stata interprets this as if you had typed

```
. tabulate lfp wc, cell miss nolabel chi2 nokey
```

Global macros are "global" because, once they are set, they can be accessed by any do-file or command until you either exit Stata or drop the macro from memory. The flip side is that global macros can be reset by any of the do-files or commands that you use along the way. By contrast, a local macro can be accessed only within the do-file in which it is defined. When the do-file terminates, the local macro disappears. We prefer using local macros whenever possible because you do not have to worry about conflicts with other do-files or commands that try to use the same macro name for a different purpose.

Local macros are defined using the `local` command, and they are referenced by placing the name of the local macro in single quotes, for example, `'myoptions'`. The two single quote marks use different symbols. On many keyboards, the left single quote ``` is in the upper left-hand corner, whereas the right single quote `'` is next to the Enter key. If the operations we just performed were in a do-file, we could have produced the same output with the following lines:

```
. local myoptions ", cell miss nolabel chi2 nokey"
. tabulate lfp wc `myoptions'
(output omitted)
```

Macros can also be used as a shorthand way to refer to lists of variables. For example, you could use these commands to create lists of variables:

```
. local demogvars "age white female"
. local edvars "highsch college graddeg"
```

Then when you run regression models, you could use the command

```
. regress y `demogvars' `edvars'
```

which Stata would translate into

```
. regress y age white female highsch college graddeg
```

Or you could use the command

```
. regress y `demogvars' `edvars' x1 x2 x3
```

which Stata would translate into

```
. regress y age white female highsch college graddeg x1 x2 x3
```


This technique has several advantages. First, it is easier to write the commands because you do not have to retype a long list of variables. Second, if you change the set of demographic variables that you want to use, you have to do it in only one place, which reduces the chance of errors.

Often, when you use a local macro name for a list of variables, the list becomes longer than one line. As with other Stata commands that extend over one line, you can use `///`, as in

```
local vars age age squared income education female occupation dadeduc ///
dadocc momeduc momocc
```

You can also define macros to equal the result of computations. After typing `local four = 2+2`, the value 4 will be substituted for ``four'`. Stata contains many macro functions in which items retrieved from memory are assigned to macros (see [P] **macro**). For example, to display the variable label that you have assigned to the variable `wc`, you can type

```
. local wclabel : variable label wc
. display "`wclabel'"
Wife College: 1=yes 0=no
```

We have only scratched the surface of the potential of macros. Macros are immensely flexible and are indispensable for a variety of advanced tasks in data analysis. By the time you finish this book, you will have mastered their use. For users interested in advanced applications, read [P] **macro**.

2.16 Loops using *foreach* and *forvalues*

Loops let you execute a set of commands multiple times. Suppose we have a four-category ordinal variable `y` with values from 1 to 4. We want to create the binary variables `y_lt2`, `y_lt3`, and `y_lt4` that equal 1 if `y` is less than the indicated value or equal 0 otherwise. We could create the variables with three **generate** commands:

```
generate y_lt2 = y<2 if y<.
generate y_lt3 = y<3 if y<.
generate y_lt4 = y<4 if y<.
```

The `if` condition selects cases where `y` is not missing. The same thing can be done with a **foreach** loop:

```
1) foreach cutpt in 2 3 4 {
2)   generate y_lt`cutpt' = y<`cutpt' if y<.
3) }
```

Line 1 starts the loop with the **foreach** command. `cutpt` is the name of a local macro that will hold the cutpoint used to dichotomize `y`. Each time through the loop, the value of `cutpt` changes, where `in` signals the start of a list of values that will be assigned in sequence to the local macro `cutpt`. The numbers 2 3 4 are the values to be assigned

to `cutpt`. The curly brace `{` indicates that the list has ended. Line 2 is the command we want to execute multiple times. Notice how the `generate` command is constructed using the macro `cutpt` created in line 1. Line 3 ends the `foreach` loop with `}`.

Here is what happens when the loop is executed. The first time through `foreach`, the local macro `cutpt` is assigned the first value in the list. This is equivalent to the command `local cutpt = 2`. Next, the `generate` command is run, where ``cutpt'` is replaced by the value assigned to `cutpt`. Thus line 2 is evaluated as

```
generate y_lt2 = y<2 if y<.
```

Next, the closing brace `}` is encountered, which sends us back to the `foreach` command in line 1. In the second pass, `foreach` assigns `cutpt` to the second value in the list, which means that the `generate` command is evaluated as

```
generate y_lt3 = y<3 if y<.
```

This continues once more, assigning `cutpt` to 4. When the `foreach` loop ends, three variables have been generated.

Next, we want to estimate binary logits on `y_lt2`, `y_lt3`, and `y_lt4`.³ We assign the independent variables to the local `rhs` (which stands for “right-hand side” of the model):

```
local rhs "yr89 male white age ed prst"
```

To run the logits, type

```
logit y_lt2 `rhs'
logit y_lt3 `rhs'
logit y_lt4 `rhs'
```

Or we could do the same thing with a loop:

```
foreach lhs in y_lt2 y_lt3 y_lt4 {
    logit `lhs' `rhs'
}
```

Using `foreach` to fit three models is probably more trouble than it is worth. But suppose that we also want to compute the frequency distribution of the dependent variable and fit a probit model. We need to add only two lines to the loop:

```
foreach lhs in y_lt2 y_lt3 y_lt4 {
    tabulate `lhs'
    logit   `lhs' `rhs'
    probit  `lhs' `rhs'
}
```

If we want to add the `missing` option to `tabulate`, we have to make the change in only one place to apply it to all three outcomes.

3. Essentially, we are making an informal assessment of the parallel regression assumption that is considered in chapter 7.

The `forvalues` command loops through numbers. The syntax is

```
forvalues local-name = range {
    commands referring to `local-name'
}
```

where *range* can be specified as

Syntax	Meaning	Example	Generates
<code>#1(#d)#2</code>	From #1 to #2 in steps of #d	<code>1(2)10</code>	1, 3, 5, 7, 9
<code>#1/#2</code>	From #1 to #2 in steps of 1	<code>1/10</code>	1, 2, 3, ..., 10
<code>#1 #t to #2</code>	From #1 to #2 in steps of (<code>#t - #1</code>)	<code>1 4 to 15</code>	1, 4, 7, 10, 13

For example, to loop through ages 40 to 80 by 5s:

```
forvalues i = 40(5)80 {
```

Or to loop from 0 to 100 by 0.1:

```
forvalues i = 0(.1)100 {
```

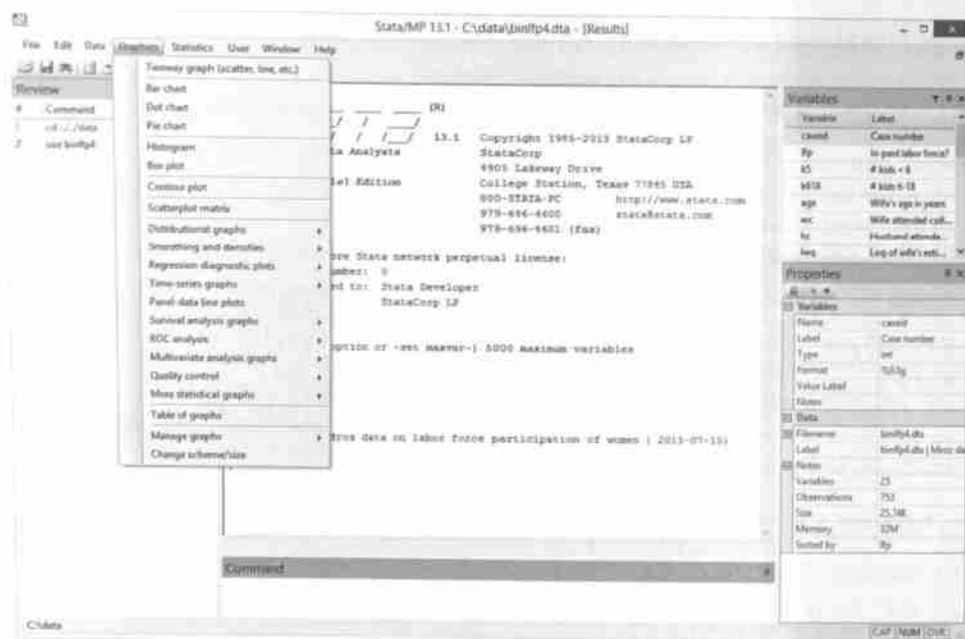
Loops are easy to use, and they make your workflow faster and more accurate. In later chapters, we use them regularly to make our work more efficient. For further information, type `help foreach` or `help forvalues`, or see [P] `foreach` or [P] `forvalues`.

2.17 Graphics

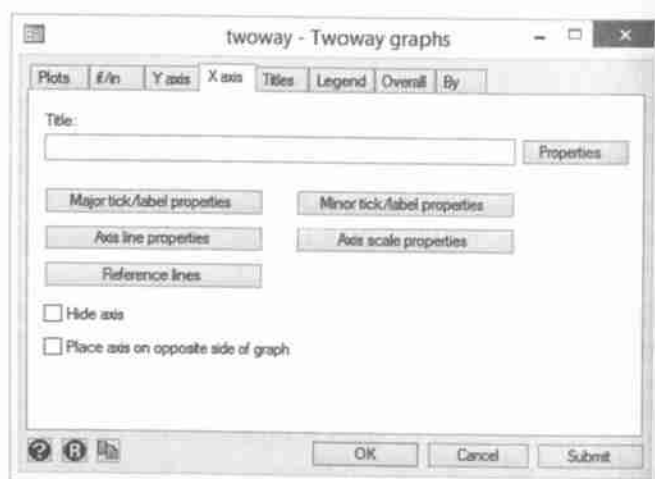
Stata has an extensive and powerful graphics system. Not only can you create many different kinds of graphs, but you have control over almost all aspects of a graph's appearance. The cost of this is that the syntax for making a graph can get complicated. Here we provide a brief introduction to graphics in Stata, focusing on the types of graphs that we use in later chapters. Our hope is to provide a basic understanding of how the graphics system works so that you can start using it.

For more information, we suggest the following. The *Stata Graphics Reference Manual* is an invaluable reference when you already have a good understanding of what you want to do, but we find it less helpful when you want to be reminded of what an option is called or get ideas about what kinds of graph to use. For this, we find Mitchell's (2012b) *A Visual Guide to Stata Graphics* to be useful. This book shows hundreds of graphs along with the Stata commands used to generate them. The book is organized in a way that makes it easy to scan the pictures until you see a graph that does what you want. You can then look at the text to find out which options to use.

The way we use Stata to make graphs differs from how we use Stata to fit models (or to do virtually anything else). Namely, when making graphs, we extensively use dialog boxes. If you pull down the **Graphics** menu (or press Alt-g), you will see a list of the plot types and families of plot types available in Stata:



Selecting any of these graph types opens a dialog box. Here we select **Twoway graph** and then the **X axis** tab, which displays the following:




You can make selections from each tab and then click on **Submit** or **OK**. **Submit** leaves the dialog box open in case you need to make additional changes, whereas **OK** closes it before generating the graph. The dialog box translates your options into the commands needed to draw the graph. These commands are echoed to the Results window, while the graph appears in a Graph window. Next, we tweak the options until

we have the graph we want. PageUp brings the **graph** command into the Command window, where you can edit it. Then, we copy the command and paste it into a do-file so that we can reproduce the graph later.

In the rest of this section, we describe the basic syntax for Stata graphics, because it is helpful to understand how this syntax works even if you ultimately use dialog boxes to do the bulk of the work. We focus on plots of one or more outcomes against a single explanatory variable, and we use the command **graph twoway**, which has the syntax

graph twoway plotype ...

(The underlined letters represent the minimum abbreviation you can type for Stata to recognize the command.) The *Stata Graphics Reference Manual* lists over 40 plot types for the **graph twoway** command. Because we discuss only the types **scatter** and **connected** here, interested readers are encouraged to consult the *Stata Graphics Reference Manual* or to type **help graph** for more information.

Graphs that you create are drawn in their own window, which should appear in front of the other Stata windows. You can resize the Graph window by clicking on and dragging the borders. If the Graph window is hidden, you can bring it to the front by clicking on .

2.17.1 The graph command

The type of graph that we use most often in this book shows how the predicted probability of observing a given outcome changes as a continuous variable changes over a specified range. For example, in chapter 6, we examine how the probability of a woman being in the labor force depends on her age and her family's income. In that chapter, we show you how to compute these predictions, but for now you can simply load them into memory with the command **use lfprgraph4, clear**. The variable **income** is family income measured in thousands of dollars, excluding any contribution made by the woman of the household. The next three variables contain the predicted probabilities of being in the labor force for women between ages 30 and 39 (**agecat1pr1**), 40 and 49 (**agecat2pr1**), or 50 and older (**agecat3pr1**):

```
. use lfprgraph4, clear
(lfprgraph4.dta | Sample predictions to plot | 2014-01-23)
. codebook income agecat1pr1 agecat2pr1 agecat3pr1, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
income	11	11	50	0	100	Family income excluding...
agecat1pr1	11	11	.4591184	.1230226	.8236541	ages 30 to 39
agecat2pr1	11	11	.3418133	.0697281	.7139289	ages 40 to 49
agecat3pr1	11	11	.234009	.0375723	.5651833	ages 50 and older

Because there are only 11 values, we can easily list them:

```
. list income agecat1pr1 agecat2pr1 agecat3pr1
```

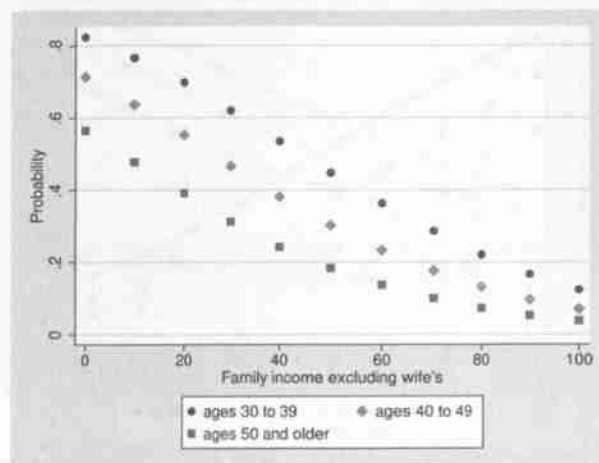
	income	age-1pr1	age-2pr1	age-3pr1
1.	0	.8236541	.7139289	.5651833
2.	10	.7668772	.6373779	.4779353
3.	20	.6985115	.5531632	.3920132
4.	30	.6200306	.4657831	.3122976
5.	40	.5347281	.3804536	.2423312
6.	50	.4473447	.3019213	.1838493
7.	60	.3630971	.2334903	.1369302
8.	70	.2864909	.1766445	.1005104
9.	80	.2204527	.1312684	.0729585
10.	90	.1660933	.0961867	.0525181
11.	100	.1230226	.0697281	.0375723

We see that as annual income increases, the predicted probability of being in the labor force decreases. Looking across rows, we see that for a given level of income, the probability of being in the labor force decreases with age. We want to display these patterns graphically.

The command `graph twoway scatter` will draw a scatterplot in which the values of one or more y variables are plotted against the values of a single x variable. Here, `income` is the x variable, and the predicted probabilities `agecat1pr1`, `agecat2pr1`, and `agecat3pr1` are the y variables. Thus for each value of x , we have three values of y . When making scatterplots with `graph twoway scatter`, the y variables are listed first, and the x variable is listed last. If we type

```
. graph twoway scatter agecat1pr1 agecat2pr1 agecat3pr1 income,
> ytitle(Probability)
```


we obtain the following graph:



The scatterplot shows the pattern of decreasing probabilities as income or age increases.

Although our simple command produces a reasonable first graph, we can make a more effective graph by adding options. We focus below on adding lines, adding titles, and improving the labels for the axes, but many other options are available. A slightly more detailed syntax for `graph twoway` is

```
graph twoway plot1 [plot2]...[plotN] [if] [in] [, twoway-options]
```

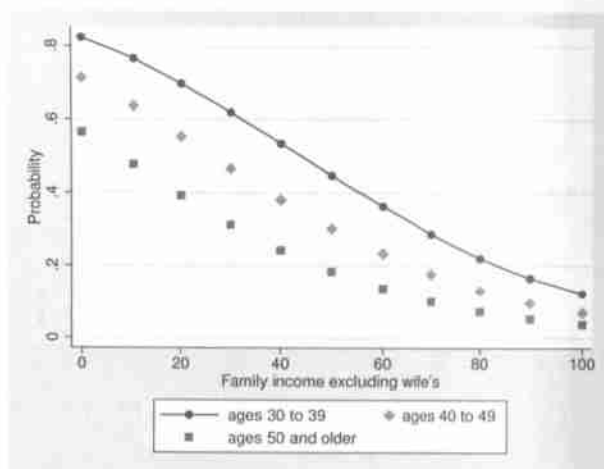
where $plot_i$ is defined to be

```
[ ( ) plottype varlist, [title("string") subtitle("string") ytitle("string")
  xtitle("string") caption("string") xlabel(values) ylabel(values)
  other-options] [ ) ]
```

This syntax highlights that it is possible to put multiple plots in the same graph, where the parentheses at the beginning and the end are used to separate the different plots when there are multiple plots. When there is only one plot, those parentheses are not required. The combined plots can be of different plot types. For instance, suppose that we want the symbols in the plot corresponding to “ages 30 to 39” to be connected. This plot type is called `connected`. For example,

```
. graph twoway (connected agecat1pr1 income)
> (scatter agecat2pr1 agecat3pr1 income), ytitle(Probability)
```


produces the following graph:



The default choices for the symbols, line styles, etc., are all quite nice. Stata made these choices within the context of an overall look, or what Stata calls a scheme. For example, because our book is published in monochrome, we want our graphs to be drawn in monochrome. Accordingly, we place

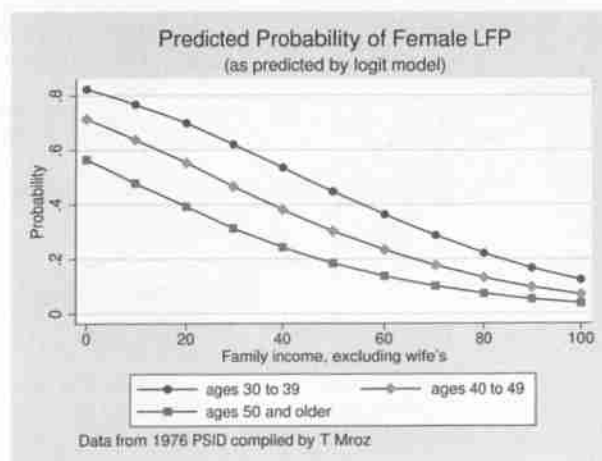
```
. set scheme s2manual
```

at the top of our do-files. This scheme makes the graph monochrome. For color graphs, we often use the `s2color` scheme. Type `help schemes` in Stata for the latest information about available schemes.

Adding titles

Next, we show how to add an overall title, a subtitle, *x*-axis and *y*-axis titles, and a caption. The following command adds titles to our graph and also replaces the scatter plot type with the connected type.

```
. graph twoway connected agecat1pr1 agecat2pr1 agecat3pr1 income,
> title("Predicted Probability of Female LFP")
> subtitle("(as predicted by logit model)")
> ytitle("Probability") xtitle("Family income, excluding wife's")
> caption("Data from 1976 PSID compiled by T Mroz")
```

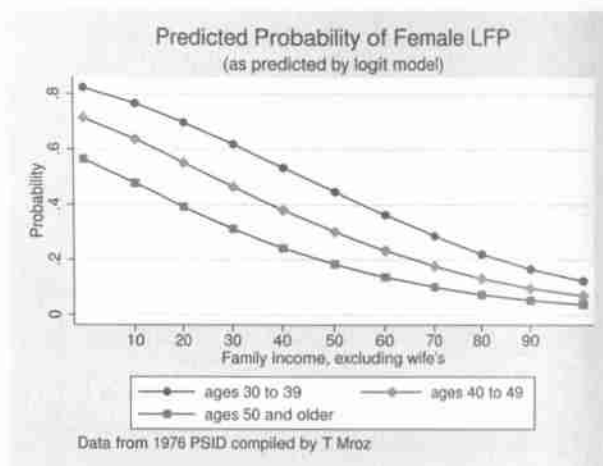



This graph is much more effective in illustrating how the probability of a woman being in the labor force declines as family income increases and with age, with the caption documenting the source of the data.

Labeling tick marks on the axes

Even though the default labels for the tick marks on our graph are reasonable, it is common to want to change them. The `ylabel()` and `xlabel()` options allow users to specify either a rule or a set of values for the tick marks. A rule in this case is simply a compact way to specify a list of values. Let's first consider specifying a list of values. Suppose that we want to restrict the range on the x axis to be from 10 to 90. We make this change by listing the values of the tick marks with `xlabel()`:

```
. graph twoway connected agecat1pr1 agecat2pr1 agecat3pr1 income,
> title("Predicted Probability of Female LFP")
> subtitle("(as predicted by logit model)")
> ylabel(0 .2 .4 .6 .8) xtitle("Family income, excluding wife's")
> caption("Data from 1976 PSID compiled by T Mroz")
> xlabel(10 20 30 40 50 60 70 80 90)
```

We could have obtained the same graph by specifying a rule for a new set of x -axis values. Although there are several ways to specify a rule, we find the form `#min(#gap)#max` most useful. In this form, the user specifies `#min` for the beginning of the sequence of values, `#gap` for the increment between each value, and `#max` for the maximum value. For instance,

```
xlabel(10(10)90)
```

specifies the same tick marks as the more cumbersome

```
xlabel(10 20 30 40 50 60 70 80 90)
```

Type `help axis_label_options` for other ways to specify a rule.

Saving graphs in memory by naming them

When you create a graph, it is displayed in a Graph window and is also saved in memory. If you close the Graph window, you can redisplay the graph with the command `graph display`. By default, a graph is stored in memory with the name `Graph`, and this graph is overwritten whenever you generate a new graph. If you want to store more than one graph in memory, you need to use the `name()` option. For example,

```
scatter y x, name(example1, replace)
```

stores the scatterplot for y against x in memory with the name `example1`, where the `replace` option indicates that you want to replace the graph named `example1` if it already exists. Then

```
scatter z x, name(example2, replace)
```

will save the scatterplot for z against x with the name `example2`.

Because Stata displays each named graph in its own window, multiple Graph windows can be displayed simultaneously. For example,

```
graph display example1
graph display example2
```

When multiple graphs have been saved, you can combine them into a single graph as discussed below.

Saving graphs to a file in .gph format

When a graph is stored in memory, it remains there until you exit Stata or erase the graph from memory. Accordingly, you are likely to want to save your graphs in a file. Specifying `saving(filename, replace)` saves the graph to a file in the working directory using Stata's proprietary format with the suffix `.gph`. Including `replace` tells Stata to overwrite a file with that name if it already exists.

Export graphs to a file in other formats

You will probably also want to export graphs to other formats that can be displayed by other programs or included in documents. This is done with the `graph export` command:

```
graph export newfilename.suffix [, replace options]
```

suffix determines the graph format. With the `replace` option, `graph export` will overwrite a graph of the same name if it already exists, which is useful in do-files. For example,

```
graph export filename.emf, replace
```

saves the graph as a Windows Enhanced Metafile (EMF), which works well with most word processors. The command

```
graph export filename.eps, replace
```

saves the graph as an Encapsulated Postscript (EPS) file, which is commonly used with \LaTeX or \TeX . See [G-2] **graph export** for a list of all the formats to which you can export graphs.


If a graph is already saved in `.gph` format and is no longer in memory, you can export it to other formats in two steps. First, redisplay the graph with the command `graph use filename`. Second, export the graph with the command `graph export filename.suffix`.

Displaying previously drawn graphs

Several commands can manipulate graphs that have been previously drawn and saved to memory or disk. `graph dir` lists graphs saved in memory or in a `.gph` file

in the working directory. `graph use` copies into memory a graph stored in a file and displays it. `graph display` redisplay a graph stored in memory.

Printing graphs

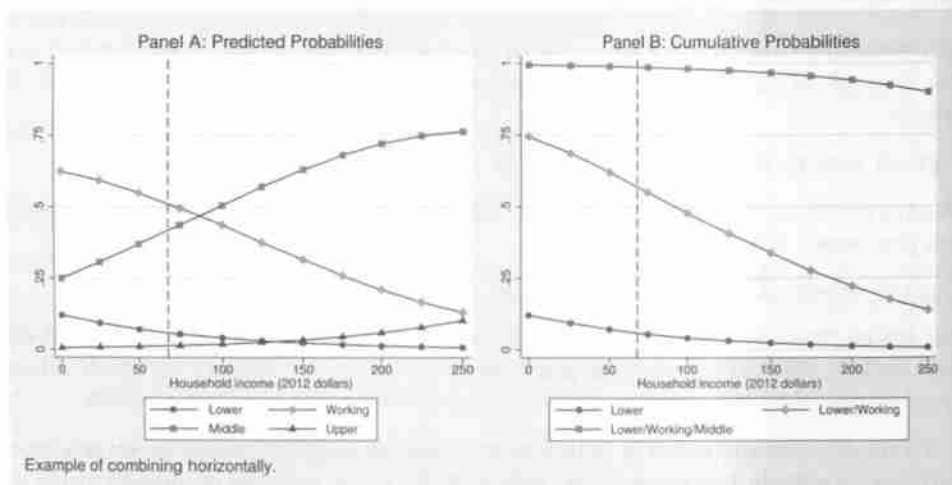
It is easiest to print a graph once it is in the Graph window. When a graph is in the Graph window, you can print it by selecting **File**→**Print**→**Graph(graphname)** from the menus or by clicking on . You can also print a graph in the Graph window with the command `graph print`. To print a graph saved to memory or a file, first use `graph use` or `graph display` to redisplay it, and then print it with the command `graph print`.

Combining graphs

Multiple graphs that are in memory can be combined. This is useful, for example, when you want to place two graphs side by side or stack them. To illustrate this, we combine two graphs from chapter 7 (see section 7.14 for details). When we created these graphs, we saved them in memory under the names `panelA` and `panelB`. We use `graph combine` to put the two graphs side by side.

```
. graph combine panelA panelB, xsize(8) ysize(4)
> caption("Example of combining horizontally.")
```

The resulting graph looks like this:

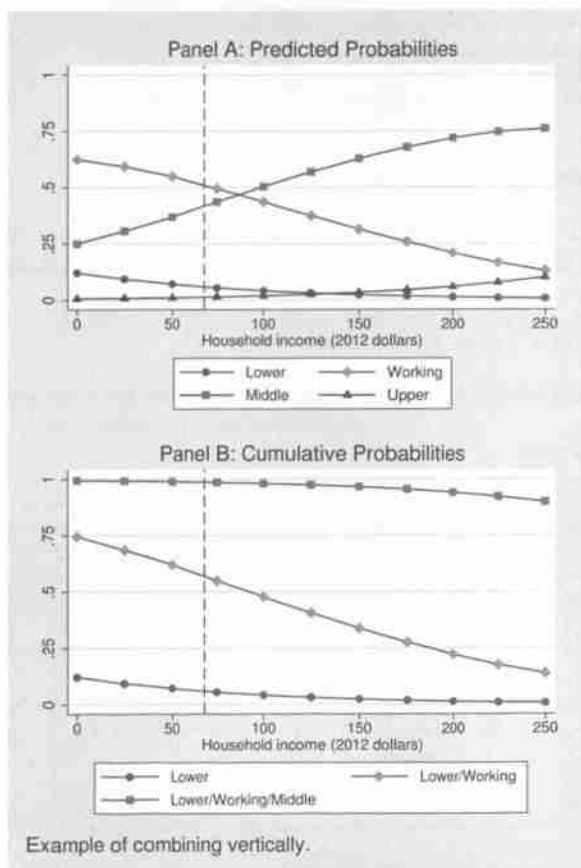


We used the options `xsize(8)` and `ysize(4)` to set the aspect ratio that we wanted. When combining graphs, you will likely need to experiment with these options to get things to look the way you want. We also used the `caption()` option to illustrate that graph options can be used with `graph combine` to customize the look of the new graph.

We could stack the graphs vertically with the `cols(1)` option, which specifies that we want a single column of graphs. We also change the aspect ratio:

```
. graph combine panelA panelB, col(1) xsize(4) ysize(6)
> caption("Example of combining vertically.")
```

Now our graph looks like this:



The *Stata Graphics Reference Manual* describes how almost any part of the combined graph can be changed.

2.18 A brief tutorial

This tutorial uses the `science4.dta` dataset that is available from the book's website. You can use your own dataset as you work through this tutorial, but you will need to change some of the commands to correspond to the variables in your data. In addition to our tutorial, the *Stata User's Guide* provides a wealth of information for new users.

Opening a log

The first step is to open a log file for recording your results. Remember that all commands are case sensitive. The commands you should type into Stata are listed with a period in front, but you do not type the period:

```
. capture log close
. log using tutorial, text
```

```
log: d:\spostdata\tutorial.log
log type: text
opened on: 24 Jan 2014, 10:12:50
```

Loading and examining the dataset

We assume that `science4.dta` is in your working directory. `clear` tells Stata to delete any existing data from memory before loading the new dataset:

```
. use science4, clear
(Long's scientific career data | 2013-10-25)
```

Next, we get descriptive statistics and variable labels for all the variables:

```
. codebook, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
<code>cit1</code>	308	50	11.60714	0	137	Citations in PhD yrs -1 to 1
<code>cit3</code>	308	57	14.97078	0	196	Citations in PhD yrs 1 to 3
<code>cit6</code>	308	65	18.37013	0	143	Citations in PhD yrs 4 to 6
<code>cit9</code>	308	74	21.07143	0	214	Citations in PhD yrs 7 to 9
<code>enrol</code>	278	9	5.564748	3	14	Years from BA to PhD
<code>faculty</code>	302	2	.5298013	0	1	Is a faculty member?
<i>(output omitted)</i>						
<code>totpub</code>	308	46	11.86364	0	84	Total publications in 9 yrs s...
<code>work</code>	302	5	2.062914	1	5	Type of first job
<code>workadm</code>	302	2	.089404	0	1	Job is in administration?
<code>worktch</code>	302	2	.615894	0	1	Job is teaching?
<code>workuniv</code>	302	2	.705298	0	1	Job is in a university?

Examining individual variables

A series of commands gives us information about individual variables. You can use whichever command you prefer or use all of them.


```
. summarize work
```

Variable	Obs	Mean	Std. Dev.	Min	Max
work	302	2.062914	1.37829	1	5

```
. tabulate work, missing
```

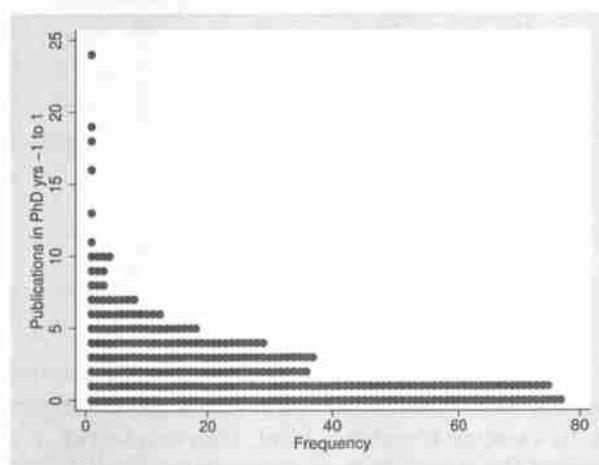
Type of first job	Freq.	Percent	Cum.
university fac	160	51.95	51.95
research univ	53	17.21	69.16
college fac	26	8.44	77.60
industry	36	11.69	89.29
administration	27	8.77	98.05
.	6	1.95	100.00
Total	308	100.00	

Graphing variables

Graphs are also useful for examining data. The command

```
. dotplot pub1
```

creates the following distribution of publications:



To save the above graph as a Windows Enhanced Metafile, type

```
. graph export tutorial_dotplot.emf, replace
(file d:\spostdata\tutorial_dotplot.emf written in Enhanced Metafile
> format)
```


Creating a binary variable

Now let's make a binary variable with faculty in universities coded 1 and all others coded 0. The command `gen isfac = (work==1) if work<.` generates `isfac` as a binary variable where `isfac` equals 1 if `work` is 1 and equals 0 otherwise. The statement `if work<.` ensures that missing values are kept as missing in the new variable.

```
. generate isfac = (work==1) if work<.
(6 missing values generated)
```

Six missing values were generated because `work` contained six missing observations.

Checking transformations

One way to check transformations is with a table. In general, it is best to look at the missing values, which requires the `missing` option:

```
. tabulate isfac work, missing
```

isfac	Type of first job					Total
	universit	research	college f	industry	administr	
0	0	53	26	36	27	142
1	160	0	0	0	0	160
.	0	0	0	0	0	6
Total	160	53	26	36	27	308

isfac	Type of first job	Total
	.	
0	0	142
1	0	160
.	6	6
Total	6	308

Labeling variables and values

For many of the regression commands, value labels for the dependent variable are essential. We start by creating a variable label, then create `isfac` to store the value labels, and finally assign the value labels to the variable `isfac`:

```
. label variable isfac "Scientist is faculty member in university"
. label define isfac 0 "NotFac" 1 "Faculty"
. label values isfac isfac
```


Then we can get labeled output:

```
. tabulate isfac
```

Scientist is faculty member in university	Freq.	Percent	Cum.
NotFac	142	47.02	47.02
Faculty	160	52.98	100.00
Total	302	100.00	

Creating an ordinal variable

The prestige of graduate programs is often referred to with the categories of adequate, good, strong, and elite. Here we create such an ordinal variable from the continuous variable for the prestige of the first job. `missing` tells Stata to show cases with missing values.

```
. tabulate job, missing
```

Prestige of 1st university job	Freq.	Percent	Cum.
1.01	1	0.32	0.32
1.2	1	0.32	0.65
1.22	1	0.32	0.97
1.32	1	0.32	1.30
(output omitted)			
4.18	2	0.65	49.03
4.42	1	0.32	49.35
4.5	6	1.95	51.30
4.69	5	1.62	52.92
.	145	47.08	100.00
Total	308	100.00	

The `recode` command makes it easy to group the categories from `job`. Of course, we then label the variable:

```
. generate jobprst = job
(145 missing values generated)
. recode jobprst . = 1/1.99=1 2/2.99=2 3/3.99=3 4/5=4
(jobprst: 162 changes made)
. label variable jobprst "Ranking of university job"
. label define prstlbl 1 "Adeq" 2 "Good" 3 "Strong" 4 "Elite"
. label values jobprst prstlbl
```

Here is the new variable (we use the `missing` option so that missing values are included in the tabulation):


```
. tabulate jobprst, missing
```

Ranking of university job	Freq.	Percent	Cum.
Adeq	31	10.06	10.06
Good	47	15.26	25.32
Strong	71	23.05	48.38
Elite	14	4.55	52.92
.	145	47.08	100.00
Total	308	100.00	

Combining variables

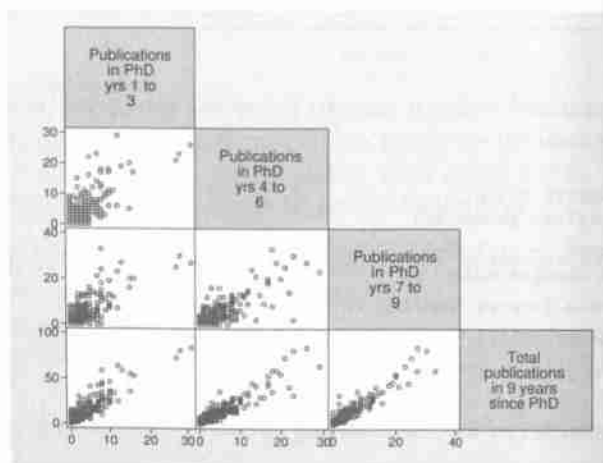
Now we create a new variable by summing existing variables. If we add `pub3`, `pub6`, and `pub9`, we can obtain the scientist's total number of publications over the 9 years since receiving a PhD.

```
. generate pubsum = pub3 + pub6 + pub9
. label variable pubsum "Total publications in 9 years since PhD"
. summarize pub3 pub6 pub9 pubsum
```

Variable	Obs	Mean	Std. Dev.	Min	Max
pub3	308	3.185065	3.908752	0	31
pub6	308	4.165584	4.780714	0	29
pub9	308	4.512987	5.315134	0	33
pubsum	308	11.86364	12.77623	0	84

A scatterplot matrix graph can be used to plot all pairs of variables simultaneously:

```
. graph matrix pub3 pub6 pub9 pubsum, half msymbol(smcircle_hollow)
```



Saving the new data

After you make changes to your dataset, save the data with a new filename. Before doing this, we add a label to the dataset and add a note that documents how the dataset was created:

```
. label data "sciwork.dta | revised science4 data | 2014-01-27"
. note _dta: "Revised by Scott Long | tutorial.do | 2014-01-27"
. save sciwork, replace
file sciwork.dta saved
```

Closing the log file

Last, we need to close the log file so that we can refer to it in the future.

```
. log close
log: d:\spostdata\tutorial.log
log type: text
closed on: 24 Jan 2014, 10:12:51
```

2.19 A do-file template

If you have read section 2.9, you know that a better idea is to create a do-file. If you download materials for this book, you can download a do-file for each chapter that repeats all the commands. The file `tutorial.do` contains the commands from this tutorial. In the do-file, we insert the `version` command after opening the log file so that our do-file will be robust to any future changes in Stata that may affect how the commands work. We also add the `exit` command to the end, which simply allows us to add additional notes if we wish to the end of the do-file without generating errors.

```
capture log close
log using tutorial, replace text
version 13.1
clear all
set linesize 80
macro drop _all
set scheme s2manual

// loading and examining the data

use science4, clear
codebook, compact

// examining individual variables

summarize work
tabulate work, missing
```



```

// graphing variables

dotplot pub1
graph export tutorial_dotplot.emf, replace

// creating a dummy variable

generate isfac = (work==1) if work<.

// checking transformations

tabulate isfac work, missing

// labeling variables and values

label variable isfac "Scientist is faculty member in university"
label define isfac 0 "NotFac" 1 "Faculty"
label values isfac isfac

tabulate isfac

// creating an ordinal variable

tabulate job, missing

generate jobprst = job
recode jobprst . = 1/1.99=1 2/2.99=2 3/3.99=3 4/5=4
label variable jobprst "Ranking of university job"
label define prstlbl 1 "Adeq" 2 "Good" 3 "Strong" 4 "Elite"
label values jobprst prstlbl

tabulate jobprst, missing

// combining variables

generate pubsum = pub3 + pub6 + pub9
label variable pubsum "Total publications in 9 years since PhD"
summarize pub3 pub6 pub9 pubsum

graph matrix pub3 pub6 pub9 pubsum, ///
    half msymbol(smcircle_hollow)
graph export tutorial_graph_matrix.emf, replace

// saving the new data

label data "sciwork.dta | revised science4 data | 2014-01-24"
note _dta: "Revised by Scott Long | tutorial.do | 2014-01-24"
save sciwork, replace

// closing the log

log close
exit

```

To execute the do-file, type `do tutorial` in the Command window or select **File**→**Do...** from the menu.

2.20 Conclusion

This chapter has provided a selective introduction to Stata that has focused on what is needed to begin work with the chapters that follow. If you are new to Stata, we obviously hope the chapter has been useful in helping you get started. In closing, we want to re-emphasize that many resources are available to help you improve your skills with Stata. As we noted, there are introductory books, websites, YouTube videos, and web forums devoted to helping you. The great payoff of Stata's elegance is that as you become comfortable with the logic of the syntax and options of the Stata commands you are using now, you will master other commands more quickly.

Data analysis is a craft, and a skilled craftsperson recognizes the value of investing in the right tools. Stata is a remarkable tool for the work we present in this book, and it is worthy of the time you invest in gaining proficiency with it. Our next step in helping you in this investment will be to provide a general introduction to using Stata to fit, evaluate, and interpret regression models. After that, we will proceed to the chapters on models for different types of categorical outcomes that comprise the heart of this book.

3 Estimation, testing, and fit

Our book deals with what we think are the most fundamental and useful cross-sectional regression models for categorical and count outcomes: binary logit and probit, ordinal logit and probit, multinomial logit, Poisson regression, and negative binomial regression. We also explore several less common models, such as the stereotype logistic regression model and the zero-inflated and zero-truncated count models. Although these models differ in many respects, they generally share common features:¹

1. Each model is fit by maximum likelihood, and many can be fit when data is collected using a complex sample survey design.
2. Hypotheses about the parameters can be tested with Wald and likelihood-ratio tests.
3. Measures of fit can be computed.
4. The models can be interpreted by examining predicted values of the outcomes, a topic that is considered in chapter 4.

Because of these similarities, the same principles and many of the same commands can be applied to each model. In this chapter, we consider the first three topics, and then we turn to issues of interpretation in chapter 4. First, we examine estimation commands, discussing how to specify models, read the results, save estimates, and create tables. We then consider statistical testing that goes beyond the routine tests of a single coefficient that are included in the output from estimation commands. This is done with the `test` and `lrtest` commands for Wald and likelihood-ratio tests. In later chapters, we present other tests of interest for a given model, such as tests of the parallel regression assumption for the ordered regression model. Finally, we consider assessing the fit of a model with scalar measures computed by our `fitstat` command and Stata's `estat` command. Later chapters focus on the application of these principles and commands to exploit the unique features of each model. This chapter and the next also serve as a reference for the syntax and options for the `SPost` commands that we introduce here and use throughout the rest of the book.

1. Many of the principles and procedures discussed in our book apply to panel models, such as those fit by Stata's `xt` and `me` commands, or models with multiple equations, such as those fit by `biprobit` or `stregress`. However, these models are not considered here.

3.1 Estimation

Each of the models we consider is fit using maximum likelihood (ML).² ML estimates are the values of the parameters that have the greatest likelihood of generating the observed sample of data if the assumptions of the model are true. To obtain the ML estimates, a likelihood function calculates how likely it is that we would observe the set of outcome values we actually observed if a given set of parameter estimates were the true parameters. For example, in linear regression with one independent variable, we need to estimate both the intercept α and the slope β . For simplicity, we are ignoring the parameter σ^2 . For any combination of possible values for α and β , the likelihood function tells us how likely it is that we would have observed the data that we did observe if these values were the population parameters. If we imagine a surface in which the range of possible values of α makes up one axis and the range of β makes up another axis, the resulting graph of the likelihood function would look like a hill; the ML estimates would be the parameter values corresponding to the top of this hill. The variance of the estimates corresponds roughly to how quickly the slope is changing near the top of the hill.

For all but the simplest models, the only way to find the maximum of the likelihood function is by numerical methods. Numerical methods are the mathematical equivalent of how you would find the top of a hill if you were blindfolded and knew only the slope of the hill at the spot where you are standing and how the slope at that spot is changing, which you could figure out by poking your foot in each direction. The search begins with start values corresponding to your location as you start your climb. From the start position, the slope of the likelihood function and the rate of change in the slope determine the next guess for the parameters. The process continues to iterate until the maximum of the likelihood function is found, called convergence, and the resulting estimates are reported. Advances in numerical methods and computing hardware have made estimation by numerical methods routine. See Cameron and Trivedi (2005), Eliason (1993), or Long (1997) for more technical discussions of ML estimation.

3.1.1 Stata's output for ML estimation

The process of iteration is reflected in the initial lines of Stata's output. Below are the first lines of the output from the logit model of labor force participation that we use as an example in chapters 5 and 6:

2. There are often convincing reasons for using Bayesian methods to fit these models. However, these methods are not generally available in Stata and hence are not considered here.


```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 agecat vc hc lvg inc
Iteration 0:   log likelihood = -514.8732
Iteration 1:   log likelihood = -453.09301
Iteration 2:   log likelihood = -452.72688
Iteration 3:   log likelihood = -452.72649
Iteration 4:   log likelihood = -452.72649
(output omitted)
```

The results begin with the iteration log, where the first line, iteration 0, reports the value of the log likelihood at the start value. Whereas earlier we talked about maximizing the likelihood function, in practice, programs maximize the log of the likelihood, which simplifies the computations and yields the same result. Here the log likelihood at the start is -514.8732 . The next four lines show the progress in maximizing the log likelihood, converging to the value of -452.72649 . The rest of the output, which is omitted here, is discussed later in this section.

3.1.2 ML and sample size

Under the usual assumptions, the ML estimator is consistent, efficient, and asymptotically normal. These properties hold as the sample size approaches infinity (see Cameron and Trivedi [2005]; Cramer [1986]; and Eliason [1993] for details). Although ML estimators are not necessarily bad estimators in small samples, the small-sample behavior of ML estimators for the models we consider is largely unknown. Except for the logit and Poisson regression, which can be fit using exact permutation methods with `exlogistic` or `expoiss`, alternative estimators with known small-sample properties are generally not available. With this in mind, Long (1997, 54) proposed the following guidelines for the use of ML in small samples:

It is risky to use ML with samples smaller than 100, while samples over 500 seem adequate. These values should be raised depending on characteristics of the model and the data. First, if there are many parameters, more observations are needed A rule of at least 10 observations per parameter seems reasonable This does not imply that a minimum of 100 is not needed if you have only two parameters. Second, if the data are ill-conditioned (for example, independent variables are highly collinear) or if there is little variation in the dependent variable (for example, nearly all the outcomes are 1), a larger sample is required. Third, some models seem to require more observations (such as the ordinal regression model or the zero-inflated count models).

3.1.3 Problems in obtaining ML estimates

Although the numerical methods used by Stata to fit models with ML are highly refined and generally work extremely well, you can encounter problems. If your sample size

is adequate, but you cannot get a solution or you get estimates that appear to not make substantive sense, one common cause is that the data have not been properly “cleaned”. In addition to mistakes in constructing variables and selecting observations, the scaling of variables can cause problems. The larger the ratio between the largest and smallest standard deviations among variables in the model, the more problems you are likely to encounter with numerical methods due to rounding. For example, if income is measured in units of \$1, income is likely to have a very large standard deviation relative to other variables. Recoding income to units of \$1,000 can solve the problem. For a detailed technical discussion of maximum likelihood estimation in Stata, see Gould, Pitblado, and Poi (2010).

Overall, however, numerical methods for ML estimation work well when your model is appropriate for your data. Still, Cramer’s (1986, 10) advice about the need for care in estimation should be taken seriously:

Check the data, check their transfer into the computer, check the actual computations (preferably by repeating at least a sample by a rival program), and always remain suspicious of the results, regardless of the appeal.

3.1.4 Syntax of estimation commands

All single-equation estimation commands that we consider in this book have the same syntax:³

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

Elements in square brackets, [], are optional. Here are a few examples for a logit model with `lfp` as the dependent variable:

```
logit lfp k5 k618 age wc lwg
logit lfp k5 k618 age wc lwg if hc == 1
logit lfp k5 k618 age wc lwg [pweight=wgtvar]
logit lfp k5 k618 age wc lwg if hc == 1, level(90)
```

You can review the output from the last estimation by typing the command name again. For example, if the most recent model that you fit was a logit model, you could have Stata replay the results by simply typing `logit`. The syntax diagram here uses 1) variable lists, 2) `if` and `in` conditions, 3) weights, and 4) options. We will discuss aspects of each of these in turn in the following sections.

3. `mlogit` is a multiple-equation estimation command, but the syntax is the same as that for single-equation commands because the independent variables are the same in all equations. The zero-inflated count models `zip` and `zinb` are the only multiple-equation commands considered in our book where different sets of independent variables can be used in each equation. Details on the syntax for these models are given in chapter 9.

3.1.5 Variable lists

depvar is the dependent variable. *indepvars* is a list of the independent variables. If no independent variables are given, a model with only the intercept is fit.

Stata automatically corrects some mistakes you may make when specifying independent variables. For example, if you include *wc* as an independent variable when the sample is restricted to a single value of *wc*, such as `logit lfp k5 age hc wc if wc==1`, then Stata drops *wc* from the model. Or suppose that you recode a *k*-category variable into a set of *k* indicator variables. Recall that one of the indicator variables must be excluded to avoid perfect collinearity. If you included all *k* indicator variables in *indepvars*, Stata automatically excludes one of them.

Using factor-variable notation in the variable list

In Stata 11 and later, you can specify a *k*-category variable as a set of indicator variables using Stata's factor-variable notation. Prefixing a variable name with *i.* tells Stata to do this. In our previous example, suppose that instead of *age* being measured in years, it was measured using three age groups with the variable *agecat*:

```
. tabulate agecat, missing
```

Wife's age group	Freq.	Percent	Cum.
30-39	298	39.58	39.58
40-49	290	38.51	78.09
50+	165	21.91	100.00
Total	753	100.00	

Variable *agecat* equals 1 for ages 30–39, 2 for 40–49, and 3 for 50 or older. If we were not using factor variables, we could recode the three categories of *agecat* to generate three dummy variables:

```
. generate age3039 = (agecat==1) if agecat < .
. label var age3039 "Age 30 to 39?"
. generate age4049 = (agecat==2) if agecat < .
. label var age4049 "Age 40 to 49?"
. generate age50plus = (agecat==3) & agecat < .
. label var age50plus "Age 50 or older?"
```

Next, we fit a model using these variables, where *age3039* is the excluded base category:

```
. logit lfp k5 k618 age4049 age50plus wc hc lwg inc, nolog
(output omitted)
```

Using factor-variable notation, we can fit the exact same model but let Stata automatically create the indicator variables:

```
. logit lfp k5 k618 i.agecat wc hc lwg inc, nolog
(output omitted)
```


Because factor-variable notation is used throughout later chapters, it is important to understand what is going on “behind the scenes” when Stata encounters `i.agecat`. Temporary variables (that is, variables that disappear after a command finishes) are generated to indicate each value of the variable that is observed in the estimation sample. These variables are named `#.varname`. For example, `i.agecat` creates `1.agecat` that is equal to 1 if `agecat` is 1 and equal to 0 for any other nonmissing value. Thus `1.agecat` is equivalent to `age3039` that was generated above. `2.agecat` is 1 if `agecat` is 2, and `3.agecat` is 1 if `agecat` is 3. By default, Stata uses the lowest value of the source variable as the base or omitted category in the model. Accordingly,

```
logit lfp k5 k618 i.agecat wc hc lwg inc, nolog
```

uses `2.agecat` and `3.agecat` as regressors, excluding `1.agecat` as the base category. If you want a different base category, specify the base with the prefix `ib#.`, where `#` is the value of the base category. In our example, to treat women ages 50 or older as our base category instead of women ages 30–39, we would specify the model as follows:

```
logit lfp k5 k618 ib3.agecat wc hc lwg inc, nolog
```

Binary, ordinal, and nominal independent variables can be treated as factor variables. A count variable or interval-level variable can also be specified as a factor variable if you do not want to impose assumptions about the form of the relationship between the explanatory variable and the outcome. For example, `i.age` would generate an indicator variable for each unique value of `age` in the sample. The values of factor variables must all be integers, and no value can be negative.

By default, any variable not specified with `i.` is treated as a continuous variable. For example, in the specification

```
logit lfp k5 k618 agecat wc hc lwg inc, nolog
```

`agecat` is treated as continuous with values 1, 2, and 3. The prefix `c.` can be used to explicitly indicate that a variable is continuous (for example, `c.inc`). In our example, the only variable where factor-variable notation is strictly necessary is `agecat`. The estimates for `inc` will be the same whether we specify the variable as `c.inc` or simply as `inc`. Coefficient estimates for binary variables will be the same whether they are included in the variable list with `i.` or not. For example, `wc` is 1 if a respondent attended college and is 0 otherwise. Whether we specify this variable as `i.wc` or simply as `wc` does not affect the estimates, although the labeling of output differs as shown below. Even so, for some of the commands we use for interpretation (discussed in chapter 4), the behavior of the commands will differ depending on whether the model was originally fit using the `i.` prefix for binary variables. Because this can lead to errors in interpretation, we suggest that you always add `i.` as a prefix for binary variables. There is no particular advantage in the above example to using `c.` for the continuous variables, because this is how the variables will be treated by default. However, we could have used factor-variable notation for all the variables in the model:

```
logit lfp c.k5 c.k618 i.agecat i.wc i.hc c.lwg c.inc, nolog
```


Specifying interaction and polynomials

Interaction terms are used to model how the coefficient for one variable differs according to values of another variable. In our example, one might hypothesize that the effect of children under age 5 on a woman's labor force participation varies depending on whether the woman has gone to college. Interactions are created as the product of two independent variables. One way to specify this is to generate a product variable and include it in the variable list, such as

```
generate wcXk5 = wc*k5
logit lfp wc k5 wcXk5 k618 age hc lwg inc, nolog
```

Alternatively, factor-variable notation allows us to put the interaction operator # between two variables in the variable list to indicate that the product of the two variables is to be included in the model. For example, instead of creating `wcXk5`, we could have used the following syntax to obtain exactly the same results:

```
logit lfp wc k5 wc#c.k5 k618 age hc lwg inc, nolog
```

Rarely do we want to include a product term without including the components of the interaction as well. The operator ## indicates that the individual variables along with their product are to be included. Thus the specification

```
logit lfp i.wc c.k5 i.wc#c.k5 k618 age hc lwg inc, nolog
```

is equivalent to

```
logit lfp i.wc##c.k5 k618 age hc lwg inc, nolog
```

In general, you should use the ## operator or be sure the individual variables are also included when using the # operator.

Notice in this example that we used factor-variable notation to make explicit that `wc` was an indicator variable (that is, we used `i.wc`) and that `k5` was continuous (that is, we used `c.k5`). If you do not indicate whether a variable is continuous or is a factor variable, Stata relies on its defaults, and these defaults depend on whether a variable is part of an interaction.

If a variable appears in an interaction, it is assumed to be a factor variable unless you use the `c.` prefix.

If a variable does not appear in an interaction term, it is considered continuous unless you use the `i.` prefix.

Moreover, the specification of a variable in an interaction supersedes the specification on the type of variable outside of the interaction. For example, if you specify a model as `logit y c.var1 c.var2 var1#var2`, then both `var1` and `var2` are treated as factor variables when creating interaction terms despite having been typed as `c.var1` and `c.var2` outside of the interaction. The safest thing when using factor-variable notation

to create interactions is to explicitly indicate how a variable is to be treated by including either the *i.* or the *c.* prefix.

The interaction notation can also be used to add polynomial terms to a model, which is a standard way of modeling nonlinear relationships between an independent variable and the outcome. Although most of the models in this book are inherently nonlinear, polynomial terms are useful for changing the way in which predictors affect the outcome. For example, in a logit model that includes age but not age-squared as a predictor, the probability of the outcome must either always increase or always decrease as age increases. But, as shown by an example in chapter 6, in some applications it makes sense for the probability to increase initially with age before decreasing at older ages. This can be done by including age, age-squared, and possibly age-cubed in the model. For example, to include age and age-squared, you could type

```
logit lfp c.age##c.age k5 k618 wc hc lwg inc, nolog
```

Because we used `##`, `c.age` is automatically included in the model. Similarly, we can use the interaction notation to include a cubed or higher-order term:

```
logit lfp k5 k618 c.age##c.age##c.age wc hc lwg inc, nolog
```

As a different example, imagine that a researcher hypothesized that the coefficients for all the other independent variables differed between women who went to college and women who did not. This could be specified as

```
logit lfp i.wc##(c.k5 c.k618 c.age i.hc c.lwg c.inc), nolog
```

where `i.wc##(c.k5 c.k618 c.age i.hc c.lwg c.inc)` is expanded to `i.wc i.wc#c.k5 i.wc#c.k618 i.wc#c.age i.wc#i.hc i.wc#c.lwg i.wc#c.inc`.

More on factor-variable notation

Factor-variable notation is extremely powerful and can save time and prevent errors in complex applications. Not only do factor variables eliminate the need to create indicator variables and polynomial terms, but when computing effects using commands such as `margins`, `marginsplot`, `mgen`, `mtable`, and `mchange`, Stata keeps track of how variables are linked. For example, if your model includes age and age-squared and you want to compute the effect of increasing age from 20 to 30 on labor force participation, with factor variables, Stata automatically makes the corresponding change in age-squared from 400 to 900. This important feature of factor-variable notation is considered more closely in the next chapter.

Still, sometimes factor variables can be confusing. First, the name of the variables automatically generated by Stata when fitting a model using factor variables might not be obvious. This is important because some postestimation commands, such as `test` and `lincom`, require the exact, symbolic name of the variable associated with a coefficient. For example, suppose that you run `logit lfp k5 k618 i.agecat i.wc i.hc lwg inc` and then want to test whether the coefficient for `wc` equals that for

hc. The command `test wc = hc` does not work because these are not the names of the factor variables included in the model. Nor does `test i.wc = i.hc` work because `i.wc` and `i.hc` tell Stata to create the indicators, but `i.wc` and `i.hc` are not the names of the factor-generated variables used in the model. The correct command is `test 1.hc = 1.wc`. To obtain the names associated with each coefficient, referred to as symbolic names, one can simply type the name of the last estimation command with the option `coeflegend`, such as `logit, coeflegend`. The model is not fit again, but the names associated with the estimates are listed. For example, fitting a model where factor-variable notation creates indicator variables and interactions produces output like this:

```
. logit lfp i.agecat c.age#c.age, nolog
(output omitted)
```

lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
agecat						
40-49	-.3667924	.3506722	-1.05	0.296	-1.054097	.3205124
50+	-.5599792	.5615575	-1.00	0.319	-1.660612	.5406533
age	.2553935	.144441	1.77	0.077	-.0277056	.5384926
c.age#c.age	-.0028917	.0016631	-1.74	0.082	-.0061513	.0003678
_cons	-4.901053	3.033286	-1.62	0.106	-10.84618	1.044078

Using the `coeflegend` option, the symbolic names are shown:

```
. logit, coeflegend
(output omitted)
```

lfp	Coef.	Legend
agecat		
40-49	-.3667924	_b[2.agecat]
50+	-.5599792	_b[3.agecat]
age	.2553935	_b[age]
c.age#c.age	-.0028917	_b[c.age#c.age]
_cons	-4.901053	_b[_cons]

Sometimes you might not be sure if your specification of the model using factor-variable notation produces the model you want. In some cases, the estimation output makes it obvious that the model is not what you intended. For example, if you mistakenly type `age##age` and obtain estimates for hundreds of interactions, you quickly realize that you meant to type `c.age##c.age`. Other mistakes are harder to catch. For example, specifying a binary predictor as `i.wc` or as `wc` produces the same estimates of parameters but leads to estimates of different quantities when using `margins` or the `m*`

commands to compute marginal effects (see section 4.5). Accordingly, it is important to check that Stata's decoding of your factor terms leads to the specification you want.

One trick that we find useful—especially with large datasets, when fitting a model can take a long time—is to use `summarize` to decode your variable list. In this example, `summarize` decodes the factor-variable notation `i.agecat` into two indicator variables and `c.age##c.age` into `age` and `age-squared`:

```
. summarize i.agecat c.age##c.age
```

Variable	Obs	Mean	Std. Dev.	Min	Max
agecat					
40-49	753	.3851262	.4869486	0	1
50+	753	.2191235	.4139274	0	1
age	753	42.53785	8.072574	30	60
c.age#c.age	753	1874.548	699.5167	900	3600

In Stata 13 and later, value labels are used to label categories of an indicator variable, such as 40-49 above. To see the values rather than the labels, you can add the `nofvlabel` option:

```
. summarize i.agecat c.age##c.age, nofvlabel
```

Variable	Obs	Mean	Std. Dev.	Min	Max
agecat					
2	753	.3851262	.4869486	0	1
3	753	.2191235	.4139274	0	1
age	753	42.53785	8.072574	30	60
c.age#c.age	753	1874.548	699.5167	900	3600

Output in regression models using factor variables was difficult to interpret prior to Stata 13 because the categories of factor variables were not labeled. For example, here is the output from Stata 12:

```
. logit lfp i.agecat i.wc i.hc k5 k618 lwg inc, nolog
```

Logistic regression

Number of obs	=	753
LR chi2(8)	=	124.30
Prob > chi2	=	0.0000
Pseudo R2	=	0.1207

Log likelihood = -452.72367

lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
agecat					
2	-.6267601	.208723	-3.00	0.003	-1.03585 - .2176705
3	-1.279078	.2597827	-4.92	0.000	-1.788242 - .7699128
1.wc	.7977136	.2291814	3.48	0.001	.3485263 1.246901

(output omitted)

And here is the much clearer output in Stata 13:

```
. logit lfp i.agecat i.wc i.hc k5 k618 lwg inc, nolog
Logistic regression      Number of obs   =       753
                        LR chi2(8)         =      124.30
                        Prob > chi2        =       0.0000
Log likelihood = -452.72367      Pseudo R2       =       0.1207
```

	lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
<i>agecat</i>							
40-49		-.6267601	.208723	-3.00	0.003	-1.03585	-.2176705
50+		-1.279078	.2597827	-4.92	0.000	-1.788242	-.7699128
<i>wc</i>							
college		.7977136	.2291814	3.48	0.001	.3485263	1.246901

(output omitted)

To take advantage of the improved labeling of factor variables in Stata 13 and later, you must assign value labels to your indicator variables, which we highly recommend.

3.1.6 Specifying the estimation sample

if and *in* restrictions can be used to define the sample of observations used to fit the model, referred to as the estimation sample, where the syntax for *if* and *in* conditions follows the guidelines in chapter 2, page 45. For example, if you want to fit a logit model only for women who went to college, you could specify `logit lfp k5 k618 age hc lwg if wc==1`.

Missing data

Estimation commands use listwise deletion to exclude cases in which there are missing values for any of the variables in the model. Accordingly, if two models are fit using the same dataset but have different sets of independent variables, it is possible to have different samples. The easiest way to understand this is with a simple example. Suppose that among the 753 cases in the sample, 23 have missing data for at least one variable. If we fit a model using all variables, we would obtain

```
. use binlfp4-missing, clear
(binlfp4-missing.dta | Mroz data with artificial missing data | 2013-10-18)

. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
Logistic regression      Number of obs   =       730
(output omitted)
```

Suppose that seven of the missing cases were missing only for `k618` and that we fit a second model excluding `k618`:

```
. logit lfp k5 i.agecat i.wc i.hc lwg inc, nolog
Logistic regression      Number of obs   =       737
(output omitted)
```


The estimation sample for the second model increased by seven cases because the seven cases with missing data for `k618` were not dropped. Thus we cannot use a likelihood-ratio test or information criteria to compare the two models (see sections 3.2 and 3.3 for details), because changes in the estimates could be due either to changes in the model specification or to the use of different samples to fit the models. When you compare coefficients across models, you want the samples to be the same. (There are other issues with comparing the coefficients of nonlinear probability models, which we will discuss in chapter 5.)

Although Stata uses listwise deletion when fitting models, this is rarely the best way to handle missing data. We recommend that you make explicit decisions about which cases to include in your analyses rather than let cases be dropped implicitly. Indeed, we would prefer that Stata issue an error rather than automatically drop cases.

The `mark` and `markout` commands make it simple to explicitly exclude missing data. `mark markvar` generates the new variable `markvar` that equals 1 for all cases. `markout markvar varlist` changes the values of `markvar` to 0 for any cases in which values of any of the variables in `varlist` are missing. The following example, where we have artificially created the missing data, illustrates how this works:

```
. use binlfp4-missing, clear
(binlfp4-missing.dta | Mroz data with artificial missing data | 2013-10-18)
. mark nomiss
. markout nomiss lfp k5 k618 agecat wc hc lwg inc
. tabulate nomiss
```

nomiss	Freq.	Percent	Cum.
0	23	3.05	3.05
1	730	96.95	100.00
Total	753	100.00	

Because `nomiss` is 1 for cases where none of the variables in our models is missing, to use the same sample when fitting both models, we add the condition `if nomiss==1`:

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc if nomiss==1, nolog
Logistic regression                               Number of obs   =       730
(output omitted)

. logit lfp k5 i.agecat i.wc i.hc lwg inc if nomiss==1, nolog
Logistic regression                               Number of obs   =       730
(output omitted)
```

Instead of selecting cases in each model with `if`, we could have used `drop if nomiss==0` to delete observations with missing values before fitting the models.

Aside: Analysis with missing data. The complex issues related to how to appropriately estimate parameters in the presence of missing data are beyond the scope of our discussion (see Little and Rubin [2002]; Enders [2010]; Allison [2001]; or the *Stata Multiple-Imputation Reference Manual*). We do want to make two brief points. First, a sophisticated but increasingly common approach to missing data involves multiple imputations, and Stata has an excellent suite of commands to make working with multiply imputed data simpler than it would be without it. What Stata does not have, at this writing, is a way of using `margins` with its multiple imputation suite. Because our primary methods of interpretation use the `margins` command, these cannot be used with multiply imputed data. Second, one way that data may be missing in a dataset is called missing at random (MAR). The name can be misleading: it does not mean missing completely at random but instead means that data are missing in a way that unbiased predictions of missing values can be made from other variables in the dataset itself. Even in the situation where missing at random data does not bias estimates (a matter outside the scope of our discussion), the techniques of interpretation that comprise much of our book involve using either the mean of a variable or the mean of an estimated effect size calculated over all observations. In either case, missing data can cause problems for computation of these averages even in cases in which the estimation of coefficients is unbiased.

Information about missing values

Although `mark` and `markout` work well for determining which observations have missing values for a set of variables, these commands do not provide information on the patterns of missing data among these variables. There are three types of questions that we might ask:

1. How many observations have no missing values? How many have missing data for one variable? For two variables? And so on.
2. What percentage of cases are missing for each variable?
3. What patterns of missing data are there among the variables? For example, do missing values on one variable tend to occur when there are missing values on some other variable?

We will consider each of these questions in turn. First, to determine the number of observations with a given number of missing values on a set of variables, the easiest way is to use the `egen` command with the `rowmiss(varlist)` function. This creates a new variable that contains the number of missing values for each observation, for which we can use `tabulate` to view the frequency distribution. For example,


```
. use gsskidvalue4, clear
(gsskidvalue4.dta | GSS 1993 & 1994 on values for kids | 2014-03-02)
. egen missing = rowmiss(age anykids black degree female kidvalue othrrace
>   year income91 income)
. tabulate missing
```

missing	Freq.	Percent	Cum.
0	2,684	58.37	58.37
1	1,709	37.17	95.54
2	195	4.24	99.78
3	7	0.15	99.93
4	3	0.07	100.00
Total	4,598	100.00	

Next, we could generate a variable, `nomiss`, that equals 1 if an observation has no missing values and 0 if an observation has any missing values: `gen nomiss = (missing==0)`. This is an alternative to using `mark` and `markout` to create an indicator variable for whether an observation has missing values for any variable in a list.

Second, if we want information on which variables have missing values, we can use `misstable summarize`.⁴ Here is an example:

```
. misstable summarize age anykids black degree female kidvalue othrrace
>   year income91 income, all showzero
```

Variable	Obs<.			Unique values	Min	Max
	Obs=.	Obs>.	Obs<.			
age	0	0	4,598	73	18	99
anykids	14	0	4,584	2	0	1
black	0	0	4,598	2	0	1
degree	14	0	4,584	5	0	4
female	0	0	4,598	2	0	1
kidvalue	1,609	0	2,989	4	1	4
othrrace	0	0	4,598	2	0	1
year	0	0	4,598	2	1993	1994
income91	0	0	4,598	24	1	99
income	495	0	4,103	21	1000	75000

The `all` option prints information for all the variables in the variable list. Otherwise, only those variables with at least one missing value are displayed. The `showzero` option displays each 0 in the table as 0 rather than as a blank space. These options make the output clearer for didactic purposes, although you may prefer to omit them in practice.

The output for `misstable summarize` distinguishes between missing values that are coded with the system missing value (.) and those coded with extended missing values (.a through .z). Some datasets use extended missing values to distinguish why different cases have missing values, like coding people who respond “don’t know” to a

4. For readers of previous editions of this book, we now use the official Stata command `misstable` instead of the `misschk` command from `SPost9`.

survey question as `.d` and coding those for whom the question is inapplicable as `.i`. In the `misstable summarize` output, the first column of numbers is the number of cases with missing values coded as the system missing value (`.`), the second is the number of cases with extended missing values, and the third is the number of cases with no missing values.

To understand the labels for these columns, you need to know that Stata considers missing values to be numerically larger than nonmissing values and that extended missing values are considered larger than the system missing value. Accordingly, `Obs<.` indicates nonmissing values and `Obs>.` indicates extended missing values that are larger than `.`, the system-missing value. In this example, we see that only four variables have missing values: `anykids` (14 cases), `degree` (14), `kidvalue` (1,609), and `income` (495).

Third, to obtain information on the patterns of missing data, we use `misstable patterns`:

```
. misstable patterns age anykids black degree female kidvalue othrrace
>      year income91 income, freq
```

Missing-value patterns
(1 means complete)

Frequency	Pattern			
	1	2	3	4
2,684	1	1	1	1
1,410	1	1	1	0
294	1	1	0	1
185	1	1	0	0
6	0	1	0	0
5	1	0	0	1
4	1	0	1	1
3	0	0	0	0
3	0	1	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
4,598				

Variables are (1) `anykids` (2) `degree` (3) `income` (4) `kidvalue`

The `freq` option displays results as frequencies rather than as percentages. Each row of the output represents a unique pattern of missing values, where 0 indicates a missing value and 1 indicates a nonmissing value. In the top row, the pattern is all 1s, showing that 2,684 cases had no missing values for any variable. In the second row, we see the most common pattern of missing values was where there were no missing values for variables 1, 2, and 3, with missing values for variable 4. At the bottom of the table, we see which number corresponds to which variable; for example, variable 1 is `anykids`. Only 4 variables are included even though 10 variables were in the variable list. This is because only variables with some missing values are included in the table.

The `misstable summarize` command has a `generate(stubname)` option to generate new variables that indicate whether observations are missing for a particular variable. For each variable that has missing data, a new variable is created whose name is `stubname` followed by the name of the variable with missing data. The new variable is assigned a value of 1 if the variable is missing for an observation and 0 if it is not missing.

These new variables could be used, for example, in a binary logit model of the sort we describe in chapters 5 and 6 using basic demographics as predictors of whether observations are missing. First, we create the indicator variables:

```
. misstable summarize age anykids black degree female kidvalue othrrace
>      year income91 income, gen(m_)
```

Variable	Obs<.			Unique values	Min	Max
	Obs=.	Obs>.	Obs<.			
anykids	14		4,584	2	0	1
degree	14		4,584	5	0	4
kidvalue	1,609		2,989	4	1	4
income	495		4,103	21	1000	75000

Then, we fit a logit model:

```
. logit m_income female black othrrace age, nolog
Logistic regression               Number of obs   =      4598
                                LR chi2(4)         =     100.88
                                Prob > chi2         =      0.0000
Log likelihood = -1520.1691       Pseudo R2      =      0.0321
```

m_income	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
female	.4777436	.1023335	4.67	0.000	.2771736	.6783135
black	.5148555	.1308433	3.93	0.000	.2584074	.7713035
othrrace	.2518131	.2355404	1.07	0.285	-.2098377	.7134639
age	.0210619	.0026597	7.92	0.000	.0158489	.0262748
_cons	-3.521194	.1612844	-21.83	0.000	-3.837306	-3.205082

Although this is only an informal assessment of the missing data, it suggests that missing values on `income` are associated with being female, black, and older.

Postestimation commands and the estimation sample

Excepting `predict`, postestimation commands for testing, assessing fit, and making predictions use the observations from the estimation sample, unless you specify otherwise. Accordingly, with these commands you do not need to worry about dropping cases that were deleted because of missing data during estimation. Stata automatically selects cases from the estimation sample by using a special variable named `e(sample)` that is created by every estimation command. This variable equals 1 for cases used to fit the

model and equals 0 otherwise. You can use this variable to select cases with `if` conditions, as in `sum female if e(sample)`. But you cannot otherwise use this variable as if it was an ordinary variable in your dataset. For example, you cannot use a command like `tabulate female e(sample)`. Instead, you need to generate an ordinary variable equal to `e(sample)`, which then allows you to tabulate the variable or do anything else with it.

If we add a variable with missing data to the model we estimated immediately above, we can see how this works:

```
. logit m_income female black othrace age i.degree, nolog
Logistic regression               Number of obs   =       4584
      (output omitted)
. generate included = e(sample)
. label var included "Cases included in logit on m_income"
. tabulate included
```

Cases included in logit on m_income	Freq.	Percent	Cum.
0	14	0.30	0.30
1	4,584	99.70	100.00
Total	4,598	100.00	

The generated variable `included` has 4,584 cases equal to 1, which is the same as the number of cases used to fit our model. The remaining 14 cases are 0, which corresponds with the 14 cases missing on the `degree` variable that we showed in the output from `misstable` above.

3.1.7 Weights and survey data

Weights indicate that some observations should be given more weight than others when computing estimates. The syntax for specifying weights is `[type=varname]`, where the square brackets are part of the command, `type` is the type of weight to be used, and `varname` is the variable containing the weights. Stata recognizes four types of weights:

1. **fweights** (frequency weights) indicate that an observation represents multiple observations with identical values. For example, if an observation has an **fweight** of 5, this is equivalent to having five identical, duplicate observations. If you do not include a weight modifier in your estimation command, this is equivalent to specifying `[fweight=1]`.
2. **pweights** (sampling weights) denote the inverse of the probability that the observation is included because of the sampling design. For example, if a case has a **pweight** of 1,200, that case had a 1 in 1,200 chance of being selected into the sample and in that sense represents 1,200 observations in the population.

3. **aweight**s (analytic weights) are inversely proportional to the variance of an observation. The variance of the j th observation is assumed to be σ^2/w_j , where w_j is the analytic weight. Analytic weights are used most often when observations are averages and the weights are the number of elements that gave rise to the average. For example, if each observation is the cell mean from a larger dataset, the data are heteroskedastic because the variance of the means decreases as the number of observations used to calculate them increases.
4. **iweight**s (importance weights) have no formal statistical definition. They are used by programmers to facilitate certain types of computations.

Frequency weights differ notably from the other types because a dataset that includes an **fweight** variable can be used to create a new dataset that yields equivalent results without frequency weights by simply repeating observations with duplicate values. As a result, frequency weights pose no issues for various techniques we consider in this book.

The use of weights is a complex topic, and it is easy to apply weights incorrectly. If you need to use weights, we encourage you to read the discussions in [U] **11.1.6 weight** and [U] **20.23 Weighted estimation**. Winship and Radbill (1994) have an accessible introduction to weights in the linear regression model. Heeringa, West, and Berglund (2010) provide an in-depth treatment along with examples using Stata in their excellent book on complex survey design, a topic we consider next.

Complex survey designs

Complex survey designs have three major features. First, samples can be divided into strata within which observations are selected separately. For example, a sample might be stratified by region of the country so that the researchers can achieve precisely the number of respondents they want from each region.

Second, samples can use clustering in which higher levels of aggregation, called primary sampling units, are selected first and then individuals are sampled from within these clusters. A survey of adolescents might use schools as its primary sampling unit and then sample students from within each school. Observations within clusters often share similarities leading to violations of the assumption of independent observations. Accordingly, when there is clustering, the usual standard errors will be incorrect because they do not adjust for the lack of independence.

Third, individuals can have different probabilities of selection. For example, the design might oversample minority populations. Such oversampling allows more precise estimates of subgroup characteristics, but probability weights must be used to obtain accurate estimates for the population as a whole.

Stata's **svy** commands for samples with complex survey designs (see the *Stata Survey Data Manual* for details) provide estimates where the standard errors are adjusted for stratification, clustering, and weights. If the sample design involves weights or clustering, but not stratification, then models can be fit using standard regression commands

with the `[pweight=weight-variable]` and `vce(cluster cluster-variable)` options. The results will be identical to those from `svy`. If the sample design involves stratification, then `svy` commands must be used to get the correct standard errors.

The first step in using commands for complex samples is to specify the key features of the survey design by using the `svyset` command. To illustrate this, we use an example from Heeringa, West, and Berglund (2010) based on the Health and Retirement Study, a representative sample of American adults over age 50.⁵ The primary sampling units (PSUs), another name for clusters, are defined by the variable `secu`, the strata by the variable `stratum`, and the probability weights by the variable `kwgtr`. We use `svyset` as follows:

```
. use svyhrs4, clear
(svyhrs4.dta) Health and Retirement Study 2006 | 2014-03-04
. svyset secu [pweight=kwgtr], strata(stratum)
      pweight: kwgtr
          VCE: linearized
Single unit: missing
  Strata 1: stratum
    SU 1: secu
    FPC 1: <zero>
```

Stata will use this information automatically when a subsequent command that supports survey estimation is prefixed by `svy:`. In our example, the outcome is whether the respondent has arthritis, and the independent variables are gender, education, and age. To fit a logit model, discussed in detail in chapter 5, we begin the command with the prefix `svy:` and afterward specify the `logit` command as we otherwise would.

```
. svy: logit arthritis male i.ed3cat age
(running logit on estimation sample)
```

Survey: Logistic regression

Number of strata	=	56	Number of obs	=	18375
Number of PSUs	=	112	Population size	=	76085117
			Design df	=	56
			F(4, 53)	=	204.28
			Prob > F	=	0.0000

arthritis	Coef.	Linearized Std. Err.	t	P> t	[95% Conf. Interval]	
male	-.5771248	.0442301	-13.05	0.000	-.6657284	-.4885212
ed3cat						
12-15 years	-.2135877	.0560052	-3.81	0.000	-.3257796	-.1013957
16+ years	-.6355568	.0650174	-9.78	0.000	-.7658022	-.5053113
age	.0478163	.0021773	21.96	0.000	.0434547	.0521779
_cons	-2.324736	.1546753	-15.03	0.000	-2.634588	-2.014884

5. We thank Steve Heeringa for allowing us to use these data.

The results indicate that men are less likely than women with similar characteristics to have arthritis, that those with more education are less likely to have arthritis, and that the probability of having arthritis increases as people get older. More precise interpretation of binary models is considered in chapter 6. Two differences merit noting between the information provided in the above results and those provided when `svy:` is not used. First, the `svy` results indicate both the number of strata and the number of PSUs in the sample. Second, in addition to the sample size, the results present the population size implied by the probability weights. In this example, the size of the target population is over 76 million.

Models fit using complex survey methods pose two problems for the postestimation techniques used in this book. First, measures of fit are often based on the model's likelihood. With survey estimation, the "likelihood" on which model estimates are based is not a true likelihood (Sribney 1997), so any technique that requires a value for the likelihood is not available.⁶ Second, some methods of interpretation that we use require estimates of the standard deviation of one or more variables in the model, which are not always available with survey estimation.

Stata supports survey estimation for nearly all the models we discuss in this book. If the model you are using does not work with the `svy:` prefix, remember that nearly all regression commands in Stata allow weights and clustering; although not ideal, this is a reasonable way to proceed if the regression command you are using does not support the `svy:` prefix.

3.1.8 Options for regression models

The following options apply to most regression models. Unique options for specific models are considered in later chapters.

noconstant constrains the intercept to equal 0. For example, in a linear regression, the command `regress y x1 x2, noconstant` would fit the model $y = \beta_1 x_1 + \beta_2 x_2 + \varepsilon$.

nolog suppresses the iteration history, which shortens the output. If you use this option, which we often do, and have problems obtaining estimates, it is a good idea to refit the model without this option and with the **trace** option.

trace lets you see the values of the parameters for each step of the iteration. This can be useful for determining which variables may be causing a problem if your model has difficulty converging.

level(#) specifies the level of the confidence interval. By default, Stata provides 95% confidence intervals for estimated coefficients, meaning that the interval around the estimated $\hat{\beta}$ would capture the true value of β 95% of the time if repeated samples were drawn. **level()** allows you to specify other intervals. For example, **level(90)**

6. When robust standard errors are used without `svy` adjustments, Stata returns a "pseudolikelihood", and we use this when computing some goodness-of-fit statistics. Because goodness-of-fit measures might be considered merely heuristic anyway, we think this is reasonable.

specifies a 90% interval. You can also change the default level with the command `set level`. For example, `set level 90` specifies 90% confidence intervals.

`vce(cluster cluster-variable)` specifies that the observations are independent across the clusters that are defined by unique values of *cluster-variable* but are not necessarily independent within clusters. Specifying this option leads to robust standard errors, as discussed below, with additional corrections for the effects of clustered data. See Hosmer, Lemeshow, and Sturdivant (2013, chap. 9) for a detailed discussion of logit models with clustered data. Using `vce(cluster cluster-variable)` does not affect the coefficient estimates but can have a large impact on the standard errors.

`vce(vctype)` specifies the type of standard errors that are reported. `vce(robust)` replaces traditional standard errors with robust standard errors, which are also known as Huber, White, or sandwich standard errors. These are discussed further next, in section 3.1.9. Gould, Pitblado, and Poi (2010) provide details on how robust standard errors are computed in Stata. Robust standard errors are automatically used if the `vce(cluster cluster-variable)` option is specified, if probability weights are used, or if a model is fit using `svy`. In earlier versions of Stata, this option was simply `robust`. Option `vce(bootstrap)` estimates the variance-covariance matrix by bootstrap, which involves repeated reestimation on samples drawn with replacement from the original estimation sample. Option `vce(jackknife)` uses the jackknife method, which involves refitting the model N times, each time leaving out a single observation. Type `help vce` option for further details.

`vsquish` eliminates the blank lines in output that are inserted when factor-variable notation is used. We sometimes use `nolog` and `vsquish` in this book to save space.

3.1.9 Robust standard errors

Robust standard errors, which are computed by Stata when the `robust` option is specified, go by a variety of names, including Huber-Eicker-White, clustered, White, heteroskedasticity-consistent, HCCM, and sandwich standard errors. In the last decade, their use has become increasingly common. For example, King and Roberts (2014) conducted a survey of articles in the *American Political Science Review* and found that 66% of the articles using regression models reported robust standard errors.

Robust standard errors are considered robust in the sense that they are correct in the presence of some types of violations of the assumptions of the model. For example, if the correct model is a binary logit model but a binary probit model is fit, the model has been misspecified. The estimates obtained by fitting a logit model cannot be maximum likelihood estimates because an incorrect likelihood function is being used (that is, a logistic probability density is used instead of the correct normal density). When a model is misspecified in this way, the usual standard errors are incorrect (White 1982). For this reason, Arminger (1995) argues that robust standard errors should be broadly used. He writes: "If one keeps in mind that most researchers misspecify the model ..., it is obvious that their estimated parameters can usually be interpreted only as minimum

ignorance estimators and that the standard errors and test statistics may be far away from the correct asymptotic values, depending on the discrepancy between the assumed density and the actual density that generated the data."

In some cases, robust standard errors are likely to work quite well. If violations of the underlying model are minor, as we would argue is the case if the true model is logit and you fit a probit model, then the robust standard errors are preferred, but the differences are likely to be quite small. In our informal simulations, they are trivially different. On the other hand, if you fit a Poisson regression model (see chapter 9) in the presence of overdispersion, Cameron and Trivedi (2013, 72–80) provide convincing evidence that robust standard errors provide a more accurate assessment of statistical significance. If there is clustering in the data, robust standard errors should be used, ideally by specifying `vce(cluster cluster-variable)` or by using svy estimation.

Arguments for robust standard errors are compelling. Some argue they should be used nearly always in practice. At the same time, robust standard errors are not a general solution to problems of misspecification, and they have important limitations. Kauermann and Carroll (2001) show that even when the model is correct, robust standard errors have more sampling variability, and sometimes far more, than the usual standard errors. This is "the price that one pays to obtain consistency". These theoretical results are consistent with simulations by Long and Ervin (2000), who found that in the linear regression model robust standard errors often did worse than the usual standard errors in samples smaller than 500. They recommended using small-sample versions that can be computed in Stata for `regress` with the options `hc2` or `hc3`. Among nonlinear models, Kauermann and Carroll (2001) consider the Poisson regression model and the logit model. They showed that the loss of efficiency when using robust standard errors can be worse than that occurring in normal models. However, we are unaware of small-sample versions of robust standard errors for nonlinear models.

There is a second and potentially very serious problem. If robust standard errors are used because a model is misspecified, it is important to consider what other implications misspecification may have. Freedman (2006) is dismissive of robust standard errors for many of the models discussed in this book for this reason, writing pointedly: "It remains unclear why applied workers should care about the variance of an estimator for the wrong parameter." Cameron and Trivedi (2010, 334) note that if a model is misspecified, the inconsistency of the parameter estimate is a far more serious problem than the consistency of the standard error.

King and Roberts (2014) argue that differences between robust and classical errors are "like canaries in the coal mine, providing clear indications that your model is misspecified and your inferences are likely biased". They suggest that a comparison of robust and nonrobust standard errors should be used as an informal test of model misspecification. Researchers should try to address the specification problems that cause robust and classical standard errors to differ, instead of considering robust standard errors to have solved the problematic implications of misspecification.

We agree with the advice that researchers should not simply opt for robust standard errors without thought as to why the standard errors might differ. This is especially so when misspecification has direct implications for estimates of quantities of interest, as it often does for the models considered in this book. Recall that in models such as logit and probit, unlike in the linear regression model, excluding an independent variable that is uncorrelated with other independent variables will bias the estimates of all parameters. Here we agree with White (1980, 828), one of the authors for whom robust standard errors are sometimes named, who cautioned that robust variance estimation “does not relieve the investigator of the burden of carefully specifying his models. Instead, it is hoped that the statistics presented here will enable researchers to be even more careful in specifying and estimating econometric models.”

In the end, what to do and advise about robust standard errors has proven one of the most difficult issues for us in revising to the current edition of this book. We received different advice when we discussed the matter with people. We could have, with some justification, used robust standard errors every time we fit a model in this book. One cost of doing so would include not being able to introduce methods that require likelihoods instead of pseudolikelihoods, including most measures of model fit that we discuss and likelihood-ratio tests. These are presently widely used in practice and also are didactically useful in understanding how the models we discuss work. Consequently, we decided against eliminating these sections of the book. To avoid confusing readers by switching back and forth between different specifications, we do not use robust standard errors for most of the examples we show.

3.1.10 Reading the estimation output

Because we have already discussed the iteration log, in the following example we suppress it with the `nolog` option and consider other parts of the output from estimation commands. Although the sample output is from `logit`, our discussion applies generally to other regression models fit by maximum likelihood. We comment briefly below on changes to the estimation output for `svy` estimation. The following output from `logit` illustrates how Stata displays results from regression commands:


```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
Logistic regression
Log likelihood = -452.72367
```

Number of obs	=	753
LR chi2(8)	=	124.30
Prob > chi2	=	0.0000
Pseudo R2	=	0.1207

lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
k5	-1.391567	.1919279	-7.25	0.000	-1.767739	-1.015395
k618	-.0656678	.068314	-0.96	0.336	-.1995607	.0682251
agecat						
40-49	-.6267601	.208723	-3.00	0.003	-1.03585	-.2176705
50+	-1.279078	.2597827	-4.92	0.000	-1.788242	-.7699128
wc						
college	.7977136	.2291814	3.48	0.001	.3485263	1.246901
hc						
college	.1358895	.2054464	0.66	0.508	-.266778	.5385569
lwg	.6099096	.1507975	4.04	0.000	.314352	.9054672
inc	-.0350542	.0082718	-4.24	0.000	-.0512666	-.0188418
_cons	1.013999	.2860488	3.54	0.000	.4533539	1.574645

Header

1. Log likelihood = -452.72367 is the value of the log likelihood at convergence.
2. Number of obs is the number of observations, excluding those with missing values and those excluded with if and in conditions.
3. LR chi2(8) is the value of a likelihood-ratio chi-squared for the test of the null hypothesis that all the coefficients associated with independent variables are simultaneously equal to 0 (see page 119 for details). The number in parentheses is the degrees of freedom for the test. When robust standard errors or probability weights are used, results from a Wald test of the same null hypothesis are shown instead.
4. Prob > chi2 indicates the *p*-value.
5. Pseudo R2 is the measure of fit also known as McFadden's (1974) R^2 . Details on how this measure is computed are given on page 126.

Estimates and standard errors

1. The leftmost column lists the variables in the model, with the dependent variable at the top. The independent variables are in the same order as they were typed on the command line. The constant, labeled `_cons`, is last. With Stata 13 and later, factor variables are labeled with their value labels. For example, the indicator variable for `agecat==2` is labeled as `agecat` followed by 40-49, which is the value

label for category 2. In Stata 11 and 12, or with the `nofvlabel` option in Stata 13 and later, the indicator variable for `agecat==2` is labeled as `agecat` followed on the next line by 2. Retyping the estimation command followed by `coeflegend` will list the symbolic names of each regression parameter.

2. Column **Coef.** contains estimates of the regression coefficients.
3. Column **Std. Err.** contains the standard errors of the estimates. With the `vce(robust)` option, these are labeled **Robust Std. Err.**
4. Column **z** contains the z test equal to the estimate divided by its standard error.
5. Column **P>|z|** is the two-tailed significance level. A significance level listed as 0.000 means that $p < 0.001$. For example, $p = 0.00049$ is rounded to 0.000.
6. Column **[95% Conf. Interval]** contains the confidence interval for each estimate. Instead of testing a specific hypothesis (for example, $H_0: \beta = 0$), we can use a confidence interval that contains the true parameter with a chosen probability, known as the confidence level. For a given confidence level, the estimated upper and lower bounds define the confidence interval.

Differences in output for `svy` estimation

With `svy` estimation, the output differs, reflecting that the estimates are no longer ML estimates.

1. In addition to the sample size, an estimate of the population size is shown.
2. The likelihood-ratio test that all coefficients are 0 is replaced by an F test.
3. The pseudo- R^2 is not shown because it is based on the log likelihood.
4. t -values are shown instead of z -values.

3.1.11 Storing estimation results

Stata considers the results of a model that has just been fit to be the active estimates. After fitting a model, you can type `ereturn list` to see a summary of the information that Stata stores about the active estimates. Postestimation commands are based on the active estimates. When a new model is fit, its results become the active estimates, replacing the previous model's estimates.

The `estimates store` and `estimates save` commands preserve the active estimation results so that they can be retrieved and used even after a new model is fit. `estimates store` saves the active estimates to memory, while `estimates save` saves them to a file. Storing estimation results is extremely useful for several reasons. For one, commands like `lrtest` and `estimates table` use results from more than one model. Because only one set of estimates can be active at a time, stored estimates are the way we can refer to multiple sets of estimates. Additionally, the `margins` command, used extensively in later chapters, makes predictions based on estimates from a model that has already been fit, meaning the active estimates. For some applications, however, we

will need to overwrite the active estimates from our regression model with estimates from `margins` by using the `post` option. Once this is done, the estimates from the regression model are no longer active and need to be restored (discussed below) as the active estimates if you want to do additional postestimation analysis of the model.

After running any estimation command, the syntax is

`estimates store name`

For example, to store the estimation results with the name `logit1`, type

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lvg inc, nolog
(output omitted)
. estimates store logit1
```

`estimates restore` restores the stored estimates to memory without fitting the model again:

`estimates restore name`

After running `estimates restore`, the estimation results in memory are the same as if we had just fit the model, even though we may have fit other models in the interim. Of course, we need to be careful about changes made to the data after fitting the model, but the same caveat about not changing the data between fitting a model and postestimation analysis applies even when `estimates store` and `estimates restore` are not used.

(Advanced) Saving estimates to a file

We mark this section as advanced because most of the time we find that storing estimation results in memory is sufficient. This section can be skipped unless you have a need to store estimates to disk. The most likely case would be if you are fitting a model that takes hours to estimate, you may want to save the result to a file so you can use them later without refitting the model.

Because `estimates store` holds the estimates in memory, estimates stored in one Stata session are not available in the next. Even within a Stata session, the command `clear all` erases stored estimates. To use estimates in a later session or after clearing memory, you can use `estimates save` to save results to a disk file:

```
estimates save filename, replace
```

For example, `estimates save model1, replace` will create the file `model1.ster`.

We can load previously saved estimation results with `estimates use`:

```
estimates use filename
```

`estimates use` restores the estimates almost as if we had just fit the model, and the “almost” here is very important. As described earlier, when we fit a model, Stata creates the variable `e(sample)` to indicate which observations were used when fitting the model. Some postestimation commands need `e(sample)` to produce proper results. However, `estimates use` does not require that the data in memory are the data used to estimate the saved results. You can even run `estimates use` without data in memory. Accordingly, `estimates use` does not restore the `e(sample)` variable. Although this prevents some postestimation commands from working, this is better than having them give wrong answers because the wrong dataset is in memory.

To deal with this issue, you can reset `e(sample)`. When doing this, you are responsible for making sure that the data loaded to memory are the same as the data when the model was originally fit. Assuming the proper data are in memory, you use the `estimates esample` command to respecify the outcome and independent variables, the `if` and `in` conditions, and the weights that were used when the model was originally fit. `e(sample)` is then set accordingly. The syntax is

```
estimates esample varlist [if] [in] [weight]
```

To show how this works, we present an example that uses a command we have not yet introduced, `estat summarize`, which provides summary statistics on the estimation sample that was used to compute the active estimates. First, we fit a model and show the summary statistics for the estimation sample. Then, we save the estimates as `model1`:


```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 if wc==1, nolog
Logistic regression
```

	Number of obs	=	212
	LR chi2(1)	=	17.91
	Prob > chi2	=	0.0000
	Pseudo R2	=	0.0673

Log likelihood = -124.06386

lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
k5	-.9348133	.2347982	-3.98	0.000	-1.395009	-.4746173
_cons	1.092971	.1765354	6.19	0.000	.7469683	1.438974

```
. estat summarize
Estimation sample logit
```

	Number of obs	=	212
--	---------------	---	-----

Variable	Mean	Std. Dev.	Min	Max
lfp	.6792453	.4678714	0	1
k5	.3301887	.6634921	0	3

```
. estimates save model1, replace
(note: file model1.ster not found)
file model1.ster saved
```

(Because of the if `wc==1` condition specified with `logit`, the estimates are based on 212 cases, not the 753 cases in the entire sample.) Next, we simulate restarting Stata with the `clear all` command. We then bring the estimates we saved to a file back as the active estimates with `estimates use`, and we replay the estimates:

```
. clear all
. estimates use model1
. estimates replay
```

```
active results
```

```
Logistic regression
```

	Number of obs	=	212
	LR chi2(1)	=	17.91
	Prob > chi2	=	0.0000
	Pseudo R2	=	0.0673

Log likelihood = -124.06386

lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
k5	-.9348133	.2347982	-3.98	0.000	-1.395009	-.4746173
_cons	1.092971	.1765354	6.19	0.000	.7469683	1.438974

```
. estimates describe
Estimation results produced by
. logit lfp k5 if wc==1, nolog
```

We can see that the old estimates are now active because the `estimates replay` command displayed the same results for our model that we had before. We also ran

`estimates describe`, which displays the command used to fit the model. All of this works even though we do not have any data in memory.⁷

Even if there are data in memory when `estimates use` is run, Stata is cautious and does not presume they are the same data used to fit the model: if you load the data with `use` and type `estat summarize`, you get an error. To avoid this error, after loading the same data, we run the `estimates esample` command:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. estimates esample: lfp k5 if wc==1
. estat summarize
```

Estimation sample logit			Number of obs =		212
Variable	Mean	Std. Dev.	Min	Max	
lfp	.6792453	.4678714	0	1	
k5	.3301887	.6634921	0	3	

With `estimates esample`, we must specify the same variables and conditions as when we fit the model. Then `estat summarize` will produce the same results it did after we fit the model initially.

3.1.12 Reformatting output with estimates table

`estimates table` reformats the results from an estimation command to look more like the tables that are seen in articles and books. `estimates table` also makes it easier to move estimation results into a word processor or spreadsheet to make presentation-quality tables. We strongly recommend using this command or some other automated procedure rather than retyping results to make tables. Not only is this less tedious, but it diminishes the possibility of errors. Also, if you revise your model and used `estimates table` in your do-file, then you automatically have the corrected tables.

The syntax is

```
estimates table [ model-name1 [ model-name2 ... ] ] [ , options ]
```

where *model-name#* is the name of a model whose results were stored using `estimates store`. If *model-name#* is not specified, the estimation results in memory are used.

Here is a simple example that lets us compare estimates from similarly specified logit and probit models, a topic considered in detail in chapter 5. We start by fitting the two models and using `estimates store` to save the estimates:

7. Of course, postestimation commands that require the data, such as `margins`, will not work.


```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 1.agecat 1.wc, nolog
(output omitted)
. estimates store logit_model

. probit lfp k5 1.agecat 1.wc, nolog
(output omitted)
. estimates store probit_model
```

We combine the estimates by using `estimates table`:

```
. estimates table logit_model probit_model, b(%12.3f) t varlabel
```

Variable	logit_model	probit_model
# kids < 6	-1.351 -7.27	-0.820 -7.56
agecat 40-49	-0.624 -3.18	-0.374 -3.17
50+	-1.190 -5.27	-0.723 -5.29
wc college	0.832 4.53	0.500 4.57
Constant	0.889 5.40	0.540 5.50

legend: b/t

`estimates table` provides great flexibility for what you include in your table. Although you should check the *Stata Base Reference Manual* or type `help estimates table` for complete information, here are some of the most basic and helpful options:

`b(format)` specifies the format used to print the coefficients. For example, `b(%9.3f)` indicates the estimates are to be in a column nine characters wide with three decimal places. For more information on formats, see `help format` or the *Stata User's Guide*.

`varwidth(#)` specifies the width of the column that includes variable names and labels on the left side of the table. This is often needed when variable labels are used.

`keep(varlist)` or `drop(varlist)` specify which of the independent variables to include in or exclude from the table.

`se[(format)]`, `t[(format)]`, and `p[(format)]` request standard errors, *t* or *z* statistics, and *p*-values, respectively. By specifying the format, you can have a different number of decimal places for each statistic, for example, `b(%9.3f)` and `t(%9.2f)`.⁸

8. `estimates table` always uses *t* for the test statistic, even though, as we will see in later chapters, in most of the cases considered in the book the test statistic is a *z* statistic and not a *t* statistic. The test statistic is correctly labeled in the output for the estimation command itself, and you should edit any *t*'s that should be *z*'s in tables that you present.

star tells Stata to print one star by a coefficient if the p -value is less than 0.05, two stars if less than 0.01, and three stars if less than 0.001. The **star** option cannot be used in conjunction with the **se**, **t**, or **p** options.

varlabel requests that variable labels be used instead of variable names in the rows of the table. Prior to Stata 13, this option was named **label**.

stats(statlist) indicates that the scalar statistics in *statlist* should be included in the base of the table. In *statlist*, **N** requests the sample size, **aic** requests Akaike's information criterion, and **bic** requests the Bayesian information criterion. To determine which other statistics can be included, after you fit a model, run **ereturn list**, for example,

```
. ereturn list
scalars:
      e(r2_p) = .0830969594435258
      e(N_cds) = 0
      e(N_cdf) = 0
      e(p) = 1.14892453941e-17
      e(chi2) = 85.56879559694858
      e(df_m) = 4
      e(ll_0) = -514.8732045671461
      e(k_eq_model) = 1
      e(ll) = -472.0888067686718
```

(output omitted)

The names of any of the scalars shown by **ereturn list** can be included in *statlist*. To determine what each scalar contains, use the command **help estimation-command** (for example, **help logit**) and check the section *Stored results*.

Although **estimates table** works well for basic tables, it is not as flexible as some programs that Stata users have written. The **estout** command written by Jann (2005) and the newer **outreg** by Gallup (2012) are very powerful and flexible. To install these programs, type **search estout** or **search outreg** and follow the links. Both programs allow you to export tables in several formats that can be imported into other programs (for example, a plain-text file, a tab-delimited text file, an HTML table, or a \LaTeX file). At this writing, however, neither of these programs uses value labels to annotate variables specified with **i.** syntax.

Tip: Copy Table and Copy Table as HTML. Here is a quick way to move tables into a word processor, spreadsheet, or email. Highlight the results in the Command window, right-click on (or Command+click in Mac OS X) the highlighted area, and select **Copy Table**. This copies the results in a unique way: it uses tabs instead of spaces to separate columns and it removes blank lines. If you paste the results into a spreadsheet, the columns are preserved, and most word processors make it easy to convert tab-delimited text into a table. If you select **Copy Table as HTML**, the results can be pasted as a table into most word processors, spreadsheets, or email clients. In Stata 13, the command `putexcel` was added to allow you to easily export matrices, expressions, and stored results to an Excel file, which allows you to pass information from your commands into an Excel file. For details, type `help putexcel` or see Crow (2013, 2014).

3.2 Testing

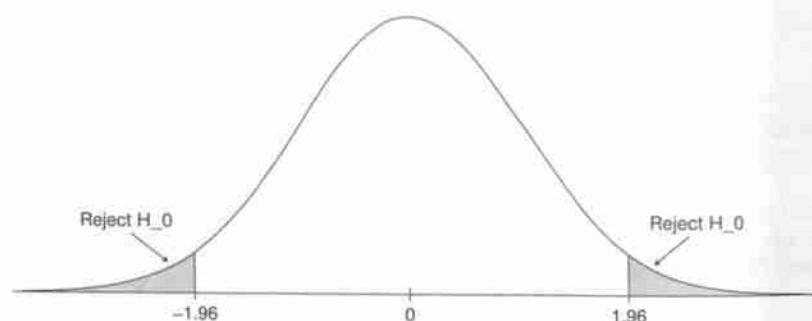
If the assumptions of the model hold, ML estimators are distributed asymptotically normally:

$$\hat{\beta}_k \stackrel{a}{\sim} \mathcal{N}(\beta_k, \sigma_{\hat{\beta}_k}^2)$$

The hypothesis $H_0 : \beta_k = \beta^*$ can be tested with the z statistic:

$$z = \frac{\hat{\beta}_k - \beta^*}{\hat{\sigma}_{\hat{\beta}_k}}$$

If H_0 is true, then z is distributed approximately normally with a mean of 0 and a variance of 1 for large samples. The sampling distribution is shown in the following figure, where the shading shows the rejection region for a two-tailed test at the 0.05 level:



For some estimators, such as linear regression implemented by `regress` and with survey estimation, the estimators have a t distribution rather than a normal distribution. The general principles of testing are, however, the same.

3.2.1 One-tailed and two-tailed tests

The probability levels in the Stata output for estimation commands are for two-tailed tests. Consider the linear regression example above for the independent variable `female`. The t statistic is -0.54 with the probability level in column `P>|t|` listed as 0.59 . This corresponds to the area of the curve that is either greater than $|t|$ or less than $-|t|$, shown by the shaded region in the tails of the sampling distribution shown in the figure above. When past research or theory suggests the sign of the coefficient, a one-tailed test might be used, and H_0 is rejected only when t or z is in the expected tail. For example, assume that my theory proposes that being a female scientist can only have a negative effect on job prestige. If we wanted a one-tailed test and the coefficient is in the expected direction (as in this example), then we want only the proportion of the distribution that is less than -0.54 , which is half of the shaded region: $0.59/2 = 0.30$. We conclude that being a female scientist does not significantly affect the prestige of the job ($t = -0.54$, $p = 0.30$ for a one-tailed test).

You should divide `P>|t|` (or `P>|z|`) by 2 only when the estimated coefficient is in the expected direction. Suppose for the effect of being female that $t = 0.54$ instead of $t = -0.54$. It would still be the case that `P>|t|` is 0.59 . The one-tailed significance level, however, would be the percentage of the distribution less than 0.54 (not less than -0.54), which is equal to $1 - (0.59/2) = 0.71$, not $0.59/2 = 0.30$. We conclude that being a female scientist does not significantly affect the prestige of the job. ($t = 0.54$, $p = 0.71$ for a one-tailed test).

Disciplines vary in their preferences for using one-tailed or two-tailed tests. Consequently, it is important to be explicit about whether p -values are for one-tailed or two-tailed tests. Unless stated otherwise, all the p -values we report in this book are for two-tailed tests.

3.2.2 Wald and likelihood-ratio tests

For models fit by ML, hypotheses can be tested with Wald tests by using `test` and with likelihood-ratio (LR) tests by using `lrtest`. Only Wald tests are available for coefficients estimated using survey estimation. For both types of tests, there is a null hypothesis H_0 that implies constraints on the model's parameters. For example, $H_0: \beta_{vc} = \beta_{hc} = 0$ hypothesizes that two of the parameters are 0 in the population.

The Wald test assesses H_0 by considering two pieces of information. First, all else being equal, the greater the distance between the estimated coefficients and the hypothesized values, the less support we have for H_0 . Second, the greater the curvature of the log-likelihood function, the more certainty we have about our estimates. This means that smaller differences between the estimates and hypothesized values are required to reject H_0 .

The LR test assesses H_0 by comparing the log likelihood from the full model that does not include the constraints implied by H_0 with a restricted model that does impose those constraints. If the constraints significantly reduce the log likelihood, then H_0 is rejected. Thus the LR test requires fitting two models.

Although the LR and Wald tests are asymptotically equivalent, they have different values in finite samples, particularly in small samples. In general, it is unclear which test is to be preferred. Cameron and Trivedi (2005, 238) review the literature and conclude that neither test is uniformly superior. Nonetheless, many statisticians prefer the LR when both are suitable. We do recommend computing only one test or the other; that is, we see no reason why you would want to compute or report both tests for a given hypothesis.

3.2.3 Wald tests with `test` and `testparm`

`test` computes Wald tests for linear hypotheses about parameters from the last model that was fit. Here we consider the most useful features of this powerful command. Features for multiple-equation models, such as `mlogit`, `zip`, and `zinb`, are discussed in chapters 8 and 9. Use `help test` for more features and `help testnl` for Wald tests of nonlinear hypotheses.

The first syntax for `test` allows you to test that one or more coefficients from the last model are simultaneously equal to 0:

```
test varlist [ , accumulate ]
```

where *varlist* contains names of independent variables from the last estimation. The `accumulate` option will be discussed shortly. Some examples of `test` after fitting the model `logit lfp k5 k618 i.agecat i.wc i.hc lwg inc` should make this first syntax clear. With one variable listed—here, `k5`—we are testing $H_0: \beta_{k5} = 0$.

```
. test k5
( 1)  [lfp]k5 = 0
      chi2( 1) =    52.57
      Prob > chi2 =    0.0000
```

The resulting chi-squared test with 1 degree of freedom equals the square of the *z* test statistic in the `logit` output. The results indicate that we can reject the null hypothesis.

If we list two variables after `test`, we can test $H_0: \beta_{k5} = \beta_{k618} = 0$:

```
. test k5 k618
( 1)  [lfp]k5 = 0
( 2)  [lfp]k618 = 0
      chi2( 2) =    52.64
      Prob > chi2 =    0.0000
```


We can reject the hypothesis that the effects of young and older children are simultaneously 0.

If we list all the regressors in the model, we can test that all the coefficients except the constant are simultaneously equal to 0. When factor-variable notation is used, variables must be specified with the *value.variable-name* syntax such as `2.agecat`. Recall that if you are not sure what name to use, you can replay the results by using the `coeflegend` option (for example, `logit, coeflegend`).

```
. test k5 k618 2.agecat 3.agecat 1.wc 1.hc lwg inc
( 1) [lfp]k5 = 0
( 2) [lfp]k618 = 0
( 3) [lfp]2.agecat = 0
( 4) [lfp]3.agecat = 0
( 5) [lfp]1.wc = 0
( 6) [lfp]1.hc = 0
( 7) [lfp]lwg = 0
( 8) [lfp]inc = 0

      chi2( 8) =    95.90
Prob > chi2 =    0.0000
```

As noted above, an LR test of the same hypothesis is part of the standard output of estimation commands, labeled as LR `chi2` in the header of the estimation output.

To test all the coefficients associated with a factor variable with more than two categories, you can use `testparm`. For example, to test that all the coefficients for `agecat` are 0, we can use `test`:

```
. test 2.agecat 3.agecat
( 1) [lfp]2.agecat = 0
( 2) [lfp]3.agecat = 0

      chi2( 2) =    24.27
Prob > chi2 =    0.0000
```

The same results are obtained with `testparm`:

```
. testparm i.agecat
( 1) [lfp]2.agecat = 0
( 2) [lfp]3.agecat = 0

      chi2( 2) =    24.27
Prob > chi2 =    0.0000
```

Because `agecat` has only two categories, the advantage of `testparm` is not great. But when there are many categories, it is much simpler to use.

The second syntax for `test` allows you to test hypotheses about linear combinations of coefficients:

```
test [exp = exp] [, accumulate]
```


For example, to test that two coefficients are equal—say, $H_0: \beta_{k5} = \beta_{k618}$:

```
. test k5=k618
( 1)  [lfp]k5 - [lfp]k618 = 0
      chi2( 1) =    45.07
      Prob > chi2 =    0.0000
```

The line labeled (1) indicates that the hypothesis $\beta_{k5} = \beta_{k618}$ has been translated to the hypothesis $\beta_{k5} - \beta_{k618} = 0$. Because the test statistic is significant, we reject the null hypothesis that the effect of having young children on labor force participation is equal to the effect of having older children.

As before, testing hypotheses involving indicator variables requires us to specify both the value and the variable. For example, to test that the coefficients for ages 40–49 versus 30–39 and for ages 50+ versus 30–39 are equal, we would use the following:

```
. test 2.agecat=3.agecat
( 1)  [lfp]2.agecat - [lfp]3.agecat = 0
      chi2( 1) =     8.86
      Prob > chi2 =    0.0029
```

The accumulate option

The `accumulate` option allows you to build more complex hypotheses based on the prior `test` command. For example, we begin with a test of $H_0: \beta_{k5} = \beta_{k618}$:

```
. test k5=k618
( 1)  [lfp]k5 - [lfp]k618 = 0
      chi2( 1) =    45.07
      Prob > chi2 =    0.0000
```

Next, add the constraint that $\beta_{wc} = \beta_{hc}$:

```
. test 1.wc=1.hc, accumulate
( 1)  [lfp]k5 - [lfp]k618 = 0
( 2)  [lfp]1.wc - [lfp]1.hc = 0
      chi2( 2) =    47.63
      Prob > chi2 =    0.0000
```

This results in a test of $H_0: \beta_{k5} = \beta_{k618}, \beta_{wc} = \beta_{hc}$. Instead of using the `accumulate` option, we could have used a single `test` command with multiple restrictions: `test (k5=k618) (1.wc=1.hc)`.

3.2.4 LR tests with `lrtest`

`lrtest` compares nested models by using an LR test. The syntax is

```
lrtest model-one [model-two]
```


where *model-one* and *model-two* are the names of estimation results stored by *estimates store*. When *model-two* is not specified, the most recent estimation results are used in its place. Typically, we begin by fitting the full or unconstrained model, and then we store the results. For example,

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
Logistic regression               Number of obs   =       753
                                LR chi2(8)          =      124.30
                                Prob > chi2          =       0.0000
                                Pseudo R2           =       0.1207

Log likelihood = -452.72367
(output omitted)
. estimates store full
```

where *full* is the name we chose for the estimation results from the full model.⁹ After we store the results, we fit a model that is nested in the full model. A nested model is one that can be created by imposing constraints on the coefficients in the prior model. Most commonly, some of the variables from the full model are excluded, which in effect constrains the coefficients for these variables to be 0. For example, if we drop *k5* and *k618* from the last model, this produces

```
. logit lfp i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. estimates store nokidvars
```

We stored the results for the nested models as *nokidvars*. Next, we compute the test:

```
. lrtest full nokidvars
Likelihood-ratio test               LR chi2(2) =      62.70
(Assumption: nokidvars nested in full) Prob > chi2 =      0.0000
```

The output indicates that *lrtest* assumes that *nokidvars* is nested in *full*. It is up to the user to ensure that the models are nested. Because our models are nested, the result is an LR test of the hypothesis $H_0: \beta_{k5} = \beta_{k618} = 0$. The significant chi-squared statistic means that we reject the null hypothesis that these two coefficients are simultaneously equal to 0. Although we fit the full model first followed by the constrained model, *lrtest* allows the constrained model to be fit first followed by the full model.

The output for all models fit by maximum likelihood includes an LR test that all the coefficients except the intercept(s) are 0. For our full model above, this is listed as *LR chi2(8) = 124.30*. The results can be computed with *lrtest* as follows:

```
. logit lfp, nolog
(output omitted)
. lrtest full .
Likelihood-ratio test               LR chi2(8) =      124.30
(Assumption: . nested in full)     Prob > chi2 =      0.0000
```

9. Although any name up to 27 characters can be used, we recommend keeping the names short but informative.

Avoiding invalid LR tests

`lrtest` does not always prevent you from computing an invalid test. There are two things that you must check: that the two models are nested and that the two models were fit using the same sample. In general, if either of these conditions is violated, the results of `lrtest` are meaningless. Although `lrtest` exits with an error message if the number of observations differs in the two models, this check does not catch those cases in which the number of observations is the same but the samples are different. One exception to the requirement of equal sample sizes is when perfect prediction removes some observations. In such case, the apparent sample sizes for nested models differ, but an LR test is still appropriate (see section 5.2.3 for details). When this occurs, the `force` option can be used to force `lrtest` to compute the seemingly invalid test. For details on ensuring the same sample size, see our discussion of `mark` and `markout` in section 3.1.6.

3.3 Measures of fit

Assessing fit involves both the analysis of the fit of individual observations and the evaluation of scalar measures of fit for the model as a whole. Regarding the former, Pregibon (1981) extended methods of residual and outlier analysis from the linear regression model to the case of binary logit and probit (see also Cook and Weisberg 1999, part IV). These measures are considered in chapter 5. Measures for count models are also available (Cameron and Trivedi 2013). Although Stata does not compute residuals and outliers for ordinal and nominal models, in some cases the tools for binary models can be used.

Many scalar measures have been developed to summarize the overall goodness of fit of regression models. A scalar measure can in some cases be useful in comparing competing models and, ultimately, in selecting a final model. Within a substantive area, measures of fit might provide a rough index of whether a model is adequate. However, there is no convincing evidence that selecting a model that maximizes the value of a given measure results in a model that is optimal in any sense other than the model's having a larger (or, in some instances, smaller) value of that measure. Measures of fit provide some information, but it is partial information that must be assessed within the context of the theory motivating the analysis, past research, and the estimated parameters of the model being considered.

3.3.1 Syntax of `fitstat`

The `SPost fitstat` command calculates many fit statistics for the estimation commands in this book. We should mention again that we often find these measures of limited utility in our own research, with the exception of the information criteria BIC and AIC. When we do use these measures, we find it helpful to compare multiple measures. `fitstat` makes this simple. The options `diff`, `saving()`, and `using()` facilitate the

comparison of measures across two models. Although *fitstat* duplicates some measures computed by other Stata commands (for example, the pseudo- R^2 in standard Stata output and the information criteria from *estat ic*), *fitstat* adds many more measures and makes it convenient to compare measures across models.

The syntax is

```
fitstat [, saving(name) using(name) ic force diff]
```

fitstat terminates with an error if the last estimation command does not return a value for the log-likelihood function for a model with only an intercept (that is, if *e(11_0)* is missing). This occurs, for example, if the *noconstant* option is used to fit a model. Although *fitstat* can be used when models are fit with weighted data, there are two limitations. First, some measures cannot be computed with some types of weights and none can be computed after *svy* estimation. Second, when *pweights* or robust standard errors are used to fit the model, *fitstat* uses the “pseudolikelihood” rather than the likelihood to compute measures of fit. Given the heuristic nature of the various measures of fit, we see no reason why the resulting measures would be inappropriate.

Options

saving(name) saves the computed measures in a matrix, *_fitstat_name*, for later comparisons. When the *saving()* option is not used, *fitstat* saves results to the matrix *_fitstat_0*.

using(name) compares the measures for the model in memory, referred to in the output as the current model, with those of the model saved as *name*.

diff compares the current model to the prior model.

ic presents only the Bayesian information criterion (BIC) and Akaike’s information criterion (AIC). When comparing two models, *fitstat* reports Raftery’s (1995) guidelines for assessing the strength of one model over another with BIC.

force is required to compare information criteria when the number of observations or the estimation method varies between the two models, or to conduct a likelihood-ratio test under circumstances in which Stata’s *lrtest* command would require the *force* option.

Models and measures

The following table shows which measures fit are computed for which models. ■ indicates that the measure is computed, and □ indicates that it is not computed.

	logit			ologit			nbreg poisson	ztab	
	regress	probit	cloglog	oprobit	mlogit	zip	zinp	slogit	mprobit
Log likelihood	■	■	■ ¹	■	■	■ ²	■	■	■
Deviance and LR χ^2	■	■	■	■	■	■	□	□	■
Deviance and Wald χ^2	□	□	□	□	□	□	■	■	□
Information measures	■	■	■	■	■	■	■	■	■
R^2 and adjusted R^2	■	□	□	□	□	□	□	□	□
Efron's R^2 and Tjur's D	□	■	■	□	□	□	□	□	□
McFadden's, ML, Cragg and Uhler's R^2	□	■	■	■	■	■	□	□	■
Count and adjusted count R^2	□	■	■	■	■	□	□	■	□
Var(e), Var(y^*), McKelvey and Zavoina's R^2	□	■	□	■	□	□	□	□	□

1: For cloglog, the log likelihood for the intercept-only model does not correspond to the first step in the iterations.

2: For zinb and zip, the log likelihood for the intercept-only model is calculated by fitting zinb or zip *depvar*, *inf(_cons)*.

3.3.2 Methods and formulas used by *fitstat*

In this section, we provide brief descriptions of each measure computed by *fitstat*. Full details for most measures along with citations to original sources are in Long (1997). We begin with formulas for several quantities that are used in the computation of other measures. We then consider the information criteria BIC and AIC. Again, these are the measures that we find most useful in practice. We then review the coefficient of determination R^2 for the linear regression model followed by numerous pseudo- R^2 's.

Quantities used in other measures

Log-likelihood based measures. Stata begins maximum likelihood iterations by computing the log likelihood of the model with all parameters but the intercept constrained to 0, referred to as $\ln L(M_{\text{Intercept}})$. The log likelihood upon convergence, referred to as $\ln L(M_{\text{Full}})$, is also listed. This information is presented in the iteration log and in the header for the estimation results.¹⁰

LR chi-square test of all coefficients. An LR test of the hypothesis that all coefficients except the intercepts are 0 can be computed by comparing the log likelihoods: $\text{LR} = 2\ln L(M_{\text{Full}}) - 2\ln L(M_{\text{Intercept}})$. LR is reported by Stata as `LR chi2(df) = #`, where the degrees of freedom in parentheses are the number of constrained parameters. For the `zip` and `zinb` models discussed in chapter 9, LR tests that the coefficients in the count portion (not the binary portion) of the model are 0.

Deviance. The deviance compares the given model with a model that has one parameter for each observation so that the model reproduces the observed data perfectly. The deviance is defined as $D = -2\ln L(M_{\text{Full}})$, where the degrees of freedom equals N minus the number of parameters. D does not have a chi-squared distribution.

Information criteria

Information measures can be used to compare both nested and nonnested models.

AIC. The formula for Akaike's information criterion (1973) used by *fitstat* and Stata's `estat ic` command is

$$\text{AIC} = -2\ln \hat{L}(M_k) + 2P_k \quad (3.1)$$

10. There are a few exceptions. For `cloglog` that we mention briefly in chapter 5, the value at iteration 0 is not the log likelihood with only the intercept. For the `zip` and `zinb` models discussed in chapter 8, the "intercept-only" model can be defined in different ways. These commands return as `e(11.0)` the value of the log likelihood with the binary portion of the model unrestricted, whereas only the intercept is free for the Poisson or negative binomial portion of the model. *fitstat* reports the value of the log likelihood from the model with only an intercept in both the binary and the count portions of the model.

where $\hat{L}(M_k)$ is the likelihood of model M_k and P_k is the number of parameters in the model (for example, $K + 1$ in the binary regression model, where K is the number of regressors). All else being equal, the model with the smaller AIC is considered the better-fitting model. Another definition of AIC is equal to the value in (3.1) divided by N . We include this quantity in the `fitstat` output as AIC divided by N .

BIC. The Bayesian information criterion (BIC) was proposed by Raftery (1995) and others as a means to compare nested and nonnested models. Because BIC imposes a greater penalty for the number of parameters in a model, it favors a simpler model compared with the AIC measure.

The BIC statistic is defined in at least three ways. Although this can be confusing, the choice of which version to use is not important, as we show after presenting the various definitions. Stata defines the BIC for model M_k as

$$\text{BIC}_k = -2 \ln \hat{L}(M_k) + \text{df}_k \ln N$$

where df_k is the number of parameters in M_k , including auxiliary parameters such as α in the negative binomial regression model. As with AIC, the smaller or more negative the BIC, the better the fit. A second definition of BIC is computed using the deviance

$$\text{BIC}_k^D = D(M_k) - \text{df}_k^D \ln N$$

where df_k is the degrees of freedom associated with the deviance. `fitstat` labels this as BIC (based on deviance). The third version, sometimes denoted as BIC' , uses the LR chi-squared with df_k' equal to the number of regressors (not parameters) in the model.

$$\text{BIC}'_k = -G^2(M_k) + \text{df}_k' \ln N$$

The difference in the BICs from two models indicates which model is preferred. Because $\text{BIC}_1 - \text{BIC}_2 = \text{BIC}'_1 - \text{BIC}'_2 = \text{BIC}_1^D - \text{BIC}_2^D$, the choice of which version of BIC to use is a matter of convenience. When $\text{BIC}_1 < \text{BIC}_2$, the first model is preferred, and accordingly, when $\text{BIC}_1 > \text{BIC}_2$, the second model is preferred. Raftery (1995) suggested these guidelines for the strength of evidence favoring M_2 against M_1 based on a difference in BIC:

Absolute difference	Evidence
0 to 2	Weak
2 to 6	Positive
6 to 10	Strong
> 10	Very strong

By default, `fitstat` shows you BIC_k , which is also computed by Stata's `estat ic`. If you specify `fitstat, ic`, then all versions of AIC and BIC are reported but non-IC measures of fit are not shown.

Example of information criteria. To compute information criteria for a single model, we fit the model and then run fitstat, saving our results with the name basemodel:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. fitstat, ic saving(basemodel)
```

		logit
AIC	AIC	923.447
	(divided by N)	1.226
BIC	BIC (df=9)	965.064
	BIC (based on deviance)	-4022.857
	BIC* (based on LRX2)	-71.307

Information criteria for a single model are not very useful. Their value comes when comparing models. Suppose we generate the variable kids, which is the sum of k5 and k618. Our new model drops k5 and k618 and adds kids. In other words, instead of a model in which the effect of an additional child age 5 or under is allowed to differ from the effect of an additional child age 6 to 18, we fit a model in which the effect of each additional child regardless of age is presumed equal. After fitting the new model, fitstat compares it with the saved model:

```
. generate kids = k5 + k618
. label var kids "Number of kids 18 or younger"
. logit lfp kids i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. fitstat, ic using(basemodel)
```

		Current	Saved	Difference
AIC	AIC	973.368	923.447	49.921
	(divided by N)	1.293	1.226	0.066
BIC	BIC (df=8/9/-1)	1010.361	965.064	45.297
	BIC (based on deviance)	-3977.561	-4022.857	45.297
	BIC* (based on LRX2)	-26.010	-71.307	45.297

Difference of 45.297 in BIC provides very strong support for saved model.

All AIC and BIC measures are smaller for the base model (listed as Saved). At the bottom of the table, it indicates that based on Raftery's criterion, there is very strong support for the saved model over the current model.

R^2 in the linear regression model

For `regress`, `fitstat` reports the coefficient of determination, which can be defined variously as

$$R^2 = 1 - \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2} = \frac{\widehat{\text{Var}}(\hat{y})}{\widehat{\text{Var}}(\hat{y}) + \widehat{\text{Var}}(\varepsilon)} = 1 - \left\{ \frac{L(M_{\text{Intercept}})}{L(M_{\text{Full}})} \right\}^{2/N} \quad (3.2)$$

The adjusted R^2 is defined as

$$\bar{R}^2 = \left(R^2 - \frac{K}{N-1} \right) \left(\frac{N-1}{N-K-1} \right)$$

where K is the number of independent variables.

Pseudo- R^2 's

Although each definition of R^2 in (3.2) gives the same numeric value in the linear regression model, each gives different values and thus provides different measures of fit when applied to other models. There are also other ways of computing measures that have some resemblance to the R^2 in the LR model. These are known as pseudo- R^2 's. Because different pseudo- R^2 measures can yield substantially different results and different software packages use different measures as their default pseudo- R^2 , when presenting results it is important to report exactly which measure is being used rather than simply saying "Pseudo- R^2 ".

McFadden's R^2 . McFadden's (1974) R^2 , also known as the LR index, compares a model with just the intercept to a model with all parameters. It is defined as

$$R_{\text{McF}}^2 = 1 - \frac{\ln \hat{L}(M_{\text{Full}})}{\ln \hat{L}(M_{\text{Intercept}})}$$

If model $M_{\text{Intercept}} = M_{\text{Full}}$, then $R_{\text{McF}}^2 = 0$, but R_{McF}^2 can never exactly equal 1. This measure is reported by Stata in the header of estimation results as **Pseudo R2** and is listed in the `fitstat` output as **R2 McFadden**. Because R_{McF}^2 always increases as variables are added to a model, an adjusted version is also available:

$$\bar{R}_{\text{McF}}^2 = 1 - \frac{\ln \hat{L}(M_{\text{Full}}) - K^*}{\ln \hat{L}(M_{\text{Intercept}})}$$

where K^* is the number of parameters, not independent variables.

Maximum likelihood R^2 . Another analogy to R^2 was suggested by Maddala (1983):

$$R_{\text{ML}}^2 = 1 - \left\{ \frac{L(M_{\text{Intercept}})}{L(M_{\text{Full}})} \right\}^{2/N}$$

This R^2 is also called the Cox-Snell (1989) R^2 .

Cragg and Uhler's R^2 . Because R^2_{ML} reaches a maximum of $1 - L(M_{Intercept})^{2/N}$, Cragg and Uhler (1970) suggested a normed measure:

$$R^2_{C\&U} = \frac{R^2_{ML}}{\max R^2_{ML}} = \frac{1 - \{L(M_{Intercept})/L(M_{Full})\}^{2/N}}{1 - L(M_{Intercept})^{2/N}}$$

This R^2 is also known as Nagelkerke's (1991) R^2 .

Efron's R^2 . For binary outcomes, Efron's (1978) pseudo- R^2 defines $\hat{\pi} = \widehat{\Pr}(y = 1 | x)$ and equals

$$R^2_{Efron} = 1 - \frac{\sum_{i=1}^N (y_i - \hat{\pi}_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2}$$

Tjur's coefficient of discrimination. For binary outcomes, Tjur (2009) motivates a goodness-of-fit measure ranging from 0 to 1 that he calls the coefficient of discrimination. D simply compares the average predicted probability when the outcome is observed as 1 to the average when the outcome is observed as 0:

$$D = \text{mean } \widehat{\Pr}(y = 1 | y = 1) - \text{mean } \widehat{\Pr}(y = 1 | y = 0)$$

The measure is a simple expression of the principle that as binary models fit better, the predicted probability of a positive outcome will increase for cases with a positive outcome and decrease for cases with a negative outcome.

$V(y^*)$, $V(\varepsilon)$, and McKelvey and Zavoina's R^2 . Some models can be defined in terms of a latent variable y^* . These include the models for binary or ordinal outcomes, such as logit, probit, ologit, and oprobit, as well as some models with censoring, such as tobit and intreg. Each model is defined in terms of a regression on a latent variable y^* :

$$y^* = x\beta + \varepsilon$$

Using $\widehat{\text{Var}}(\hat{y}^*) = \hat{\beta}' \widehat{\text{Var}}(x) \hat{\beta}$, McKelvey and Zavoina (1975) proposed

$$R^2_{M\&Z} = \frac{\widehat{\text{Var}}(\hat{y}^*)}{\widehat{\text{Var}}(y^*)} = \frac{\widehat{\text{Var}}(\hat{y}^*)}{\widehat{\text{Var}}(\hat{y}^*) + \text{Var}(\varepsilon)}$$

In models for categorical outcomes, $\text{Var}(\varepsilon)$ is assumed to identify the model.

Count and adjusted count R^2 . Observed and predicted values can be used in models with categorical outcomes to compute what is known as the count R^2 . Consider the binary case where the observed y is 0 or 1 and $\hat{\pi}_i = \widehat{\Pr}(y = 1 | x_i)$. Define the expected outcome as

$$\hat{y}_i = \begin{cases} 0 & \text{if } \hat{\pi}_i \leq 0.5 \\ 1 & \text{if } \hat{\pi}_i > 0.5 \end{cases}$$

This allows us to construct a table of observed and predicted values, such as that produced for the logit model by the command `estat classification`:

```
. estat classification
Logistic model for lfp
```

Classified	True		Total
	D	-D	
+	334	182	516
-	94	143	237
Total	428	325	753

Classified + if predicted Pr(D) >= .5
True D defined as lfp != 0

Sensitivity	Pr(+ D)	78.04%
Specificity	Pr(- -D)	44.00%
Positive predictive value	Pr(D +)	64.73%
Negative predictive value	Pr(-D -)	60.34%
False + rate for true -D	Pr(+ -D)	56.00%
False - rate for true D	Pr(- D)	21.96%
False + rate for classified +	Pr(-D +)	35.27%
False - rate for classified -	Pr(D -)	39.66%
Correctly classified		63.35%

We see that positive responses were predicted for 516 observations, of which 334 were correctly classified because the observed response was positive ($y = 1$), whereas the other 182 were incorrectly classified because the observed response was negative ($y = 0$). Likewise, of the 237 observations for which a negative response was predicted, 143 were correctly classified and 94 were incorrectly classified.

A seemingly appealing measure is the proportion of correct predictions, referred to as the count R^2 ,

$$R_{\text{Count}}^2 = \frac{1}{N} \sum_j n_{jj}$$

where the n_{jj} 's are the number of correct predictions for outcome j . The count R^2 can give a faulty impression that the model is predicting very well. In a binary model, without knowledge about the independent variables, it is possible to correctly predict at least 50% of the cases by choosing the outcome category with the largest percentage of observed cases. To adjust for the largest row marginal,

$$R_{\text{AdjCount}}^2 = \frac{\left(\sum_j n_{jj} \right) - \max_r (n_{r+})}{N - \max_r (n_{r+})}$$

where n_{r+} is the marginal for row r . The adjusted count R^2 is the proportion of correct guesses beyond the number that would be correctly guessed by choosing the largest marginal.

3.3.3 Example of fitstat

To examine all the measures of fit, we repeat our example for information criteria, but this time we use `fitstat` without the `ic` option. We fit our base model and save the `fitstat` results with the name `basemodel`:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. fitstat, saving(basemodel)
(output omitted)
```

Next, we fit a model that includes the variable `kids`, which is the sum of `k5` and `k618`, and drops `k5` and `k618`. `fitstat` compares this model with the saved model:

```
. gen kids = k5 + k618
. label var kids "Number of kids 18 or younger"
. logit lfp kids i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. fitstat, using(basemodel)
```

	Current	Saved	Difference
Log-likelihood			
Model	-478.684	-452.724	-25.960
Intercept-only	-514.873	-514.873	0.000
Chi-square			
D (df=745/744/1)	957.368	905.447	51.921
LR (df=7/8/-1)	72.378	124.299	-51.921
p-value	0.000	0.000	0.000
R2			
McFadden	0.070	0.121	-0.050
McFadden (adjusted)	0.055	0.103	-0.048
McKelvey & Zavoina	0.125	0.215	-0.090
Cox-Snell/ML	0.092	0.152	-0.061
Cragg-Uhler/Nagelkerke	0.123	0.204	-0.081
Efron	0.090	0.153	-0.063
Tjur's D	0.091	0.153	-0.063
Count	0.633	0.676	-0.042
Count (adjusted)	0.151	0.249	-0.098
IC			
AIC	973.368	923.447	49.921
AIC divided by N	1.293	1.226	0.066
BIC (df=8/9/-1)	1010.361	965.064	45.297
Variance of			
e	3.290	3.290	0.000
y-star	3.761	4.192	-0.431

Note: Likelihood-ratio test assumes current model nested in saved model.

Difference of 45.297 in BIC provides very strong support for saved model.

estat ic

estat ic lists the information criteria AIC and BIC for the last model. See page 123 for details.

estat vce

estat vce lists the variance-covariance matrix for the coefficient estimates. For further details, see **help estat vce**.

3.5 Conclusion

This concludes our discussion of the basic commands and options that are used for fitting, testing, and assessing fit. In part II of the book, we show how these commands can be applied with models for different types of outcomes. Before turning to those models, we review the methods of interpretation that are the primary focus of our book.

4 Methods of interpretation

In this chapter, we introduce methods of interpretation that will be used throughout the rest of the book. Models for categorical outcomes are nonlinear, and this nonlinearity is the fundamental challenge that must be addressed for effective interpretation. Most simply, this means that you cannot effectively interpret your model by presenting a list of the estimated parameters. Instead, we believe that the most effective way to interpret these models is by first fitting the model and then computing and examining postestimation predictions of the outcomes. Most of this chapter is a first pass at showing you how to do this.

To understand the fundamentally important point about why interpreting results from nonlinear models is more difficult, we begin with a heuristic discussion of the idea of nonlinearity before introducing commands that facilitate interpretation. Afterward, we will provide an overview of the different approaches to interpretation that will be presented in the chapter, followed by detailed discussions. We discuss at length Stata's `margins` command, which is vital to these techniques of interpretation, as well as the `m*` commands we have written for this book, which make using `margins` easier.

4.1 Comparing linear and nonlinear models

Linear models

Consider a linear regression model where y is the dependent variable, x is a continuous independent variable, and d is a binary independent variable. The model is

$$y = \alpha + \beta x + \delta d + \varepsilon$$

Given the usual assumption that $E(\varepsilon | x, d) = 0$, it follows that

$$E(y | x, d) = \alpha + \beta x + \delta d$$

which is graphed in figure 4.1. The solid line plots $E(y | x, d)$ as x changes, holding $d = 0$; that is, $E(y | x, d) = \alpha + \beta x$. The dashed line plots $E(y | x, d)$ as x changes when $d = 1$, which has the effect of changing the intercept: $E(y | x, d) = \alpha + \beta x + \delta 1 = (\alpha + \delta) + \beta x$.

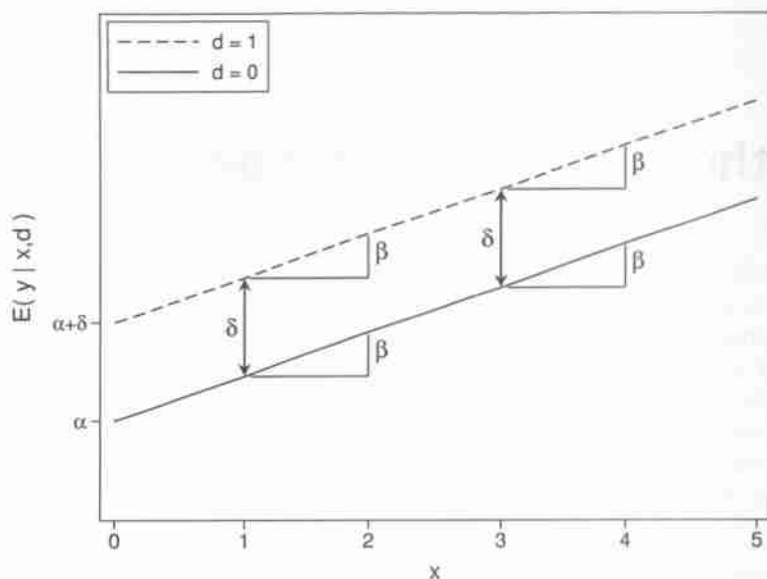


Figure 4.1. A simple linear model

The effect of x on y can be computed as the partial derivative of $E(y|x,d)$ with respect to x . This is sometimes called the marginal change:

$$\frac{\partial E(y|x,d)}{\partial x} = \frac{\partial (\alpha + \beta x + \delta d)}{\partial x} = \beta$$

The marginal change is the ratio of the change in the expected value of y to the change in x , when the change in x is infinitely small, holding d constant. In linear models, the marginal change equals the discrete change in $E(y|x,d)$ as x changes by one unit, holding other variables constant. In our notation, we indicate that x is changing by a discrete amount with Δx using $(x \rightarrow x+1)$ to indicate that x changes from its current value to be 1 larger (for example, from 10 to 11 or from 9.3 to 10.3):

$$\frac{\Delta E(y|x,d)}{\Delta x(x \rightarrow x+1)} = \{ \alpha + \beta(x+1) + \delta d \} - \{ \alpha + \beta x + \delta d \} = \beta$$

When x increases by 1, $E(y|x,d)$ increases by β regardless of the values for x and d at the point where change is measured. This is shown by the four small triangles in figure 4.1 with bases of length 1 and heights of β .

The effect of d cannot be computed as a partial derivative because d is discrete. Instead, we measure the change in $E(y|x, d)$ with a discrete change from 0 to 1 indicated as $\Delta d(0 \rightarrow 1)$:

$$\frac{\Delta E(y|x, d)}{\Delta d(0 \rightarrow 1)} = (\alpha + \beta x + \delta 1) - (\alpha + \beta x + \delta 0) = \delta$$

When d changes from 0 to 1, $E(y|x, d)$ changes by δ units regardless of the level of x . This is shown by the two arrows labeled δ in figure 4.1 marking the distance between the solid and dashed lines.

The distinguishing feature for interpretation in linear models is that the effect of a given change in an independent variable is the same regardless of the value of that variable at the start of its change and regardless of the level of the other variables in the model. Interpretation only needs to specify which variable is changing, by how much, and that other variables are being held constant. Given the simple structure of linear models, such as **regress**, most interpretations require only reporting the estimates. There are, however, important exceptions. In our discussion, we assumed that the model does not include polynomial terms such as x^2 or interactions such as xd . When such terms are included, the linear model becomes nonlinear in the sense we consider in the next section.

Nonlinear models

We use a logit model to illustrate the idea of nonlinearity. Let $y = 1$ if the outcome occurs, say, if a person is in the labor force, and otherwise $y = 0$. The curves are from the logit equation

$$\Pr(y = 1|x, d) = \frac{\exp(\alpha + \beta x + \delta d)}{1 + \exp(\alpha + \beta x + \delta d)} \quad (4.1)$$

where the α , β , and δ parameters in this equation are unrelated to those for the linear model. Once again, x is continuous and d is binary. The model is shown in figure 4.2.

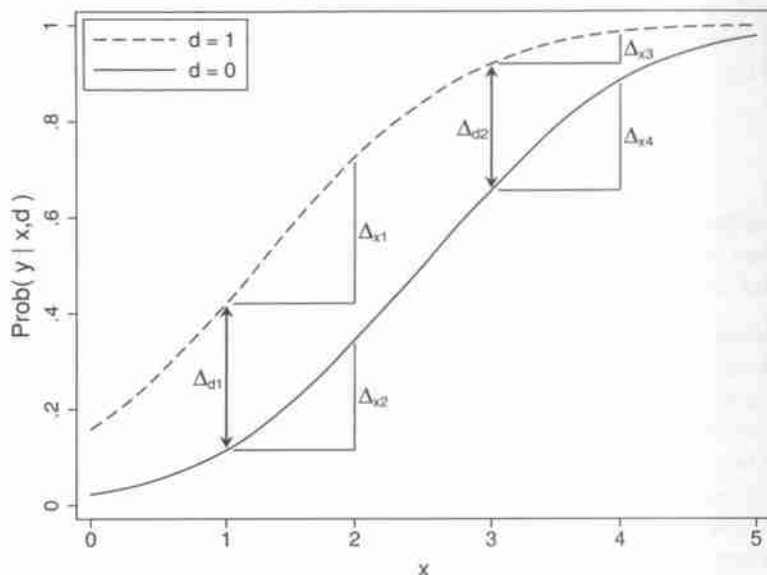


Figure 4.2. A simple nonlinear model

The nonlinearity of the model makes it more difficult to interpret the effects of x and d on the probability of y occurring. For example, neither the marginal change $\partial \Pr(y = 1 | x, d) / \partial x$ nor the discrete change $\Delta \Pr(y = 1 | x, d) / \Delta d(0 \rightarrow 1)$ are constant, but instead depend on the values of x and d . Consider the effect of changing d from 0 to 1 for a given value of x . This effect is the distance between the solid curve for $d = 0$ and the dashed curve for $d = 1$. Because the curves are not parallel, the magnitude of the difference in the predicted probability at $d = 1$ compared with $d = 0$ depends on the value of x where the difference is computed. Accordingly, $\Delta_{d1} \neq \Delta_{d2}$. Similarly, the magnitude of the effect of x depends on the values of x and d where the effect is evaluated so that $\Delta_{x1} \neq \Delta_{x2} \neq \Delta_{x3} \neq \Delta_{x4}$. In nonlinear models, the effect of a change in a variable depends on the values of all variables in the model and is no longer simply equal to a parameter of the model. Accordingly, the methods of interpretation that we recommend for nonlinear models are largely based on the use of predictions, which we consider in the next section.

4.2 Approaches to interpretation

The primary methods of interpretation presented in this book are based on predictions from the model. The model is fit and the estimated parameters are used to make predictions at values of the independent variables that are (hopefully) useful for understanding the substantive implications of the nonlinear model. These methods depend

critically on Stata's `predict` and `margins` commands, which are the foundation for the `SPost` commands `mtable`, `mchange`, and `mgen` (referred to collectively as the `m*` commands). Although the basic use of these commands is straightforward, they have many—sometimes subtle—features that are valuable for fully interpreting your model. This chapter provides an overview of general principles and syntax for these commands. Details on why you would use each feature are explained fully when the commands are used in later chapters to interpret specific models.

When reading this section the first time, you might not fully understand all the details. Indeed, our examples necessarily use models that are explained in later chapters. Be assured, however, that these will become clearer as you see the commands applied later in the book. You might even find it most effective to initially skim this chapter, returning to it as you read later chapters. These commands take time to master, but the effort pays off.

4.2.1 Method of interpretation based on predictions

We use predictions in four basic ways.

Predictions for each observation. Most fundamentally, predictions can be computed for each observation by using `predict`. Predictions include the probabilities of outcomes as well as rates for count models. We often start our analysis by examining the distribution of predictions in the estimation sample.

Predictions at specified values. Predicted values at specific values of the independent variables can be computed using the commands `margins` and `mtable`. These commands can compute predictions at substantively interesting combinations of values of the independent variables, which we refer to as profiles or ideal types. In some cases, tables of predictions are arranged by the level of one or more explanatory variables and can succinctly summarize processes affecting the outcomes.

Marginal effects. An important way to examine the effects of a variable is to compute how changes in the variable are associated with changes in the outcomes, holding other variables constant. These changes, known as marginal effects, can be computed as a marginal change when a regressor changes by an infinitely small amount or as a discrete change when a regressor changes by a fixed amount. Marginal effects are computed by `margins`, `mtable`, and `mchange`, which can easily compute average marginal effects and marginal effects at the mean.

Graphs of predictions. For continuous independent variables, graphs often effectively summarize effects. Stata's `marginsplot` elegantly plots a single outcome category based on predictions from `margins`. Just as `margins` can only compute predictions for one outcome at a time, `marginsplot` does not allow you to plot multiple outcomes. Because this is essential for models with nominal and ordinal outcomes, we wrote `mgen` to generate variables with predictions for all outcomes. These variables containing predictions can be plotted using Stata's `graph` command.

4.2.2 Method of interpretation using parameters

Although the predictions used for each of these methods are computed using the model's estimated parameters, in some cases the parameters themselves can be used for interpretation. Examples include odds ratios for binary models, standardized coefficients for latent outcomes, and factor changes in rates for count models. These are considered in detail in later chapters.

4.2.3 Stata and SPost commands for interpretation

The most fundamentally important command for sophisticated interpretation using predictions is Stata's `margins` command. This command is incredibly powerful, flexible, and general. As a consequence, it can be rather intimidating to use. To make `margins` simpler to use, we wrote a series of "wrappers" that use `margins` for their computations; they simplify the process of specifying the predictions you want and produce output that is easier to interpret. Nonetheless, there are times when you might need to use `margins`, either because our commands did not anticipate something you want to do or because we encountered technical issues that made using `margins` the only option. Accordingly, even if our `m*` commands seem to do everything you want, you should have some familiarity with what `margins` does and how it works. This will also give you a better understanding of what our commands are doing.

4.3 Predictions for each observation

The `predict` command computes predicted values for each observation in the current dataset. `predict` has many options that depend on the model that was fit. Here we consider only the options that provide information we use regularly in later chapters. If you type `help estimation-command` (for example, `help logit`), you can click on the **Also See** tab in the upper-right corner of the window and then select the `postestimation` entry for the command (for example, `[R] logit postestimation`); the `postestimation` entry includes details on how `predict` works for that estimation command.

The simplest syntax for `predict` is

```
predict newvarlist
```

where `newvarlist` contains the name or names of the variables that are generated to hold the predictions. How many variables and what is predicted depends on the model. The defaults for estimation commands used in this book are listed in the following table.

Estimation command	Default prediction
<code>regress</code>	Expected value: $E(y x) = x\hat{\beta}$
<code>logistic, logit, probit</code>	Probability: $\widehat{\Pr}(y = 1 x)$
<code>mlogit, mprobit, ologit, oprobit, slogit</code>	Probabilities: $\widehat{\Pr}(y = k x)$
<code>nbreg, poisson, tnbreg, tpoisson, zinb, zip</code>	Expected rate: $E(y x)$
	Probabilities: $\widehat{\Pr}(y = k x)$

As an example, we compute predicted probabilities for a logit model of women's labor force participation. Below, `predict` generates the variable `prob` (a name we chose) containing the probabilities of a woman being in the labor force:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. predict prob
(option pr assumed; Pr(lfp))
. summarize prob
```

Variable	Obs	Mean	Std. Dev.	Min	Max
prob	753	.5683931	.1945282	.0135618	.9512301

The summary statistics show that in the sample of 753, the probabilities range from 0.014 to 0.951 with an average of 0.568. A detailed discussion of predicted probabilities for binary models is provided in chapter 6.

For models with ordinal or nominal outcomes (chapters 7 and 8), `predict` computes the predicted probability of an observation falling into each of the outcome categories. So, instead of providing a single variable name for predictions, you specify as many names as there are categories. For example, after fitting a model for a nominal dependent variable with four categories, you can use `predict prob1 prob2 prob3 prob4`. The new variables contain the predicted probabilities of being in the first, second, third, and fourth categories.

For count models, by default `predict` computes the rate or expected count. Or, `predict newvarname, pr(#)` computes the predicted probabilities of the specified counts. For example, `predict prob0, pr(0)` generates the variable `prob0` containing estimates of $\Pr(y = 0 | x)$. And, `predict pr(#low, #high)` computes probabilities of contiguous counts. For example, `predict probto3, pr(1,3)` generates the variable `probto3` containing estimates of $\Pr(1 \leq y \leq 3 | x)$. Details are given in chapter 9.

4.4 Predictions at specified values

Interpreting predictions at substantively interesting values of the regressors is an essential method of interpretation. Such predictions can be made with `margins` and with

the `m*` commands we have written that are based upon `margins`. We focus on several aspects of these commands:

1. Specifying values of the independent variables.
2. Explaining how factor variables are handled.
3. Using a *numlist* for predictions at multiple values.
4. Making predictions by the levels of a variable defining groups.
5. Predicting quantities that are not the default for `margins`.

We will explain how to use the `margins` command to make predictions, and then we will show how the same things (and more) can be done using the `m*` command `mtable`. This section includes a lot of details on the mechanics of using these commands, many of which will be directly applicable to the `mchange` and `mgen` commands in later sections. Chapters 5 through 9 illustrate their use for interpreting models for binary, ordinal, nominal, and count models.

4.4.1 Why use the `m*` commands instead of `margins`?

Our `m*` commands `mtable`, `mchange`, and `mgen` are “wrappers” for `margins`. By wrapper, we mean that the `m*` commands translate your specification into a series of `margins` commands that actually do the computations. If all the `SPost` commands do is run `margins`, why would you use them instead of `margins`? Conversely, if you are convinced that it is more effective to use the `m*` commands, why do you need to learn more about `margins`?

Although we think `margins` is an extraordinary command, it can be difficult to use, and the output can be difficult to interpret. It does difficult things with amazing ease and also makes you work hard to do some simple things. In some ways, frankly, `margins` is more suited for a programmer than for a data analyst. For example, if you are fitting models with ordinal, nominal, or count outcomes, you have to run `margins` once for each outcome. Then, you face the tedious and error-prone task of combining the output from several `margins` commands. The learning curve for `margins` can also be steep. Our commands make it easier—sometimes much easier. The output is more compact, and if you want to plot the predictions, variables are automatically generated.

If this is so, why learn to use `margins`? First, many of the options for our commands are identical to those for `margins`. Indeed, our commands simply run `margins` and collect the results. Accordingly, what you learn about `margins` will apply exactly to our commands. Second, we only discuss the features of `margins` that we find most useful for the models in this book. You might want to use other options, and many should work with our commands.¹ This allows you to use those features while taking advantage of the convenience of the `m*` commands. Knowing something about `margins`

1. Given the many features in `margins`, there are options we have not tried. If a `margins` option does not work with one of our commands, let us know and we will consider adding it to our commands.

makes this easier. Third, if one of our commands does not work the way you expect, you should examine the results from `margins`. Each of our commands has the option `commands` to list the `margins` commands used to get our results. The `details` option will display the full output from `margins`, which can sometimes be thousands of lines of output. To understand this output, you need to know something about `margins`. And finally, `margins` can do some things that cannot be done with our commands. In such cases, as illustrated in later chapters, we rely on `margins`.

4.4.2 Using margins for predictions

The `margins` command allows you to predict many quantities and compute summary measures of your predictions. To begin, it is helpful to see how `margins` is related to `predict`. Consider the example in section 4.3, where `predict` computed the probability of labor force participation for each observation in the sample. Using `summarize` to analyze the variable generated by `predict`, we found that the mean probability was 0.568. We can obtain exactly the same mean prediction with `margins`:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. margins
Predictive margins                                Number of obs   =       753
Model VCE      : OIM
Expression     : Pr(lfp), predict()
```

	Delta-method					
	Margin	Std. Err.	z	P> z	[95% Conf. Interval]	
_cons	.5683931	.0166014	34.24	0.000	.535855	.6009312

When no options are specified, `margins` calculates the mean of the default quantity computed by `predict` for the estimation command. Earlier, we used `predict` to generate a variable with the probabilities of $y = 1$ for each observation, and we used `summarize` to compute the mean probability. Behind the scenes, this is what `margins` does.²

An advantage of using `margins` to compute the average predicted probability is that it provides the 95% confidence interval along with a test of the null hypothesis that the average prediction is 0. Stata does this using the delta method to compute standard errors (see [R] `margins` or Agresti [2013, 72]). In this example, testing that the mean prediction is 0 is not useful; but, when we later use `margins` to compute marginal effects, testing whether estimates differ from 0 is very useful.

2. Unfortunately, the variables generated by `margins` disappear when the command ends. We hope that in the future an option will be added to `margins` that allows the user to retain these variables.

In addition to computing average predictions over the sample, `margins` allows us to compute predictions at specified values of the independent variables, whether those values occur in the sample or not. The most common example of this is computing the prediction with all variables at their mean by using the `atmeans` option:

```
. margins, atmeans
Adjusted predictions      Number of obs   =      753
Model VCE      : OIM
Expression      : Pr(lfp), predict()
at              : k5              = .2377158 (mean)
                  k618            = 1.353254 (mean)
                  1.agecat         = .3957503 (mean)
                  2.agecat         = .3851262 (mean)
                  3.agecat         = .2191235 (mean)
                  0.wc             = .7184595 (mean)
                  1.wc             = .2815405 (mean)
                  0.hc             = .6082337 (mean)
                  1.hc             = .3917663 (mean)
                  lwg              = 1.097115 (mean)
                  inc              = 20.12897 (mean)
```

	Delta-method			z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.					
_cons	.5778714	.0197056		29.33	0.000	.539249	.6164937

The output begins by listing the values of the independent variables at which the prediction was calculated, called the *atlegend*, where (mean) lets you know that these values are the means.

The `at()` option for specifying values

The `at()` option allows us to set specific values of the independent variables at which predictions are calculated. Stata refers to the specification of values within `at()` as the *atspec*. As an example, we can compute the probability of labor force participation for a young woman with one young child, one older child, and so on:


```
. margins, at(k5=1 k618=1 agecat=1 wc=0 hc=0 lwg=1 inc=20)
Adjusted predictions                                Number of obs =      753
Model VCE      : OIM
Expression     : Pr(lfp), predict()
at             : k5              =      1
                k618            =      1
                agecat          =      1
                wc              =      0
                hc              =      0
                lwg             =      1
                inc             =     20
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_cons	.3694891	.0466304	7.92	0.000	.2780951	.460883

In this example, we are computing predictions for a hypothetical observation that has the values of the independent variables specified with `at()`. The output shows these values in the *atlegend* before displaying the prediction.

If we want some of the variables to be held at their means, say, `inc` and `lwg`, we could remove them from the *atspec* and include the *atmeans* option:

```
. margins, at(k5=1 k618=1 agecat=1 wc=0 hc=0) atmeans
Adjusted predictions                                Number of obs =      753
Model VCE      : OIM
Expression     : Pr(lfp), predict()
at             : k5              =      1
                k618            =      1
                agecat          =      1
                wc              =      0
                hc              =      0
                lwg             =  1.097115 (mean)
                inc             = 20.12897 (mean)
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_cons	.3823232	.0475022	8.05	0.000	.2892206	.4754259

With *atmeans*, all variables not in the *atspec* are set equal to their means.

There are two other ways we could have done the same thing. (With *margins*, most things can be done multiple ways!) First, we could enter the values for the means in the *atspec*:

```
margins, at(k5=1 k618=1 agecat=1 wc=0 hc=0 lwg=1.097 inc=20.13)
```

This is not exactly the same because the specified values of means were rounded. Second, we can use the (*atstat*) suboption within *at()*:

```
margins, at(k5=1 k618=1 agecat=1 wc=0 hc=0 (mean) lwg inc)
```


We can also specify other statistics. For example, we can calculate the predicted probability with *lwg* held at its mean by using (*mean*) and *inc* held at its median by using (*median*).

```
. margins, at(k5=1 k618=1 agecat=1 wc=0 hc=0 (mean) lwg (median) inc)
Adjusted predictions                                Number of obs   =       753
Model VCE      : OIM
Expression     : Pr(1fp), predict()
at              : k5              =           1
                  k618           =           1
                  agecat          =           1
                  wc              =           0
                  hc              =           0
                  lwg             =    1.097115 (mean)
                  inc             =    17.7 (median)
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_cons	.4026216	.0472423	8.52	0.000	.3100284	.4952148

For continuous predictors, *atstat* can be *mean*, *median*, *p#* for percentiles from 1 to 99, *min* for the minimum, and *max* for the maximum. If you try to use these options for variables specified as factor variables using *i.*, an error is generated.

asobserved for average predictions

When we do not specify values for the independent variables, either using *atmeans* or *at()*, the *margins* command computes the mean of the predictions across observations. Average predictions, which are sometimes called as-observed predictions, are the default. You can make the default explicit with the command *margins, asobserved*. In linear models, *atmeans* and *asobserved* predictions are identical, but because they differ in nonlinear models, it is important to understand why they differ. The substantive implications of this difference are particularly important when computing marginal effects, discussed briefly in section 4.5 and in detail in section 6.2.

If you do not set the value for an independent variable in the *atspec* or with *atmeans*, the variable is treated as-observed. For example, in the command

```
margins, at(k5=1 k618=1 agecat=1 wc=0 hc=0 lwg=1.097)
```

the variable *inc* is treated as-observed because it is not included in the *atspec*. We can make this explicit by using (*asobserved*) *inc*:

```
margins, at(k5=1 k618=1 agecat=1 wc=0 hc=0 lwg=1.097 (asobserved) inc)
```

Values for as-observed variables are not listed in the output because they vary across observations. For example, here *inc* is an as-observed variable, so it is not shown in the *atlegend*:


```
. margins, at(k5=1 k618=1 agecat=1 wc=0 hc=0 lwg=1)
Predictive margins                                Number of obs   =       753
Model VCE      : OIM
Expression     : Pr(lfp), predict()
at             : k5               =         1
                  k618            =         1
                  agecat           =         1
                  wc               =         0
                  hc               =         0
                  lwg              =         1
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_cons	.374379	.044939	8.33	0.000	.2863001	.4624579

To understand what happens with the as-observed variable `inc`, we show how to use a series of Stata commands to compute the same prediction. First, for each variable specified in `at()`, we replace the observed value with the specified value:

```
. replace k5 = 1
(635 real changes made)
. replace k618 = 1
(568 real changes made)
. replace agecat = 1
(455 real changes made)
. replace wc = 0
(212 real changes made)
. replace hc = 0
(295 real changes made)
. replace lwg = 1
(753 real changes made)
```

The observed values of all variables except `inc` have been replaced. For every observation in the dataset, `k5` is 1, `k618` is 1, and so on. Variable `inc` has been left “as observed”. Next, we make predictions with the observed values of `inc` and the changed values of other variables:

```
. predict prob
(option pr assumed; Pr(lfp))
. label var prob "predict with fixing values of all but inc"
```

Computing the mean of the predictions,

```
. sum prob
```

Variable	Obs	Mean	Std. Dev.	Min	Max
prob	753	.374379	.0804292	.0392212	.5418246

we obtain the same value as `margins` when `inc` was treated as-observed. Nontrivially, although the mean prediction is the same, we have not computed the standard error of the prediction, which `margins` provides.

Predictions using interaction and polynomial terms

In section 3.1.5, we showed how factor-variable notation allows you to specify interaction terms (for example, `i.wc##c.age`) and quadratic terms (for example, `c.age##c.age`). When factor-variable notation is used in the `atspec`, `margins` handles these terms properly. For example, let's say your specification includes `i.wc##c.age`. In this case, when `margins`, `at(age=30 wc=1)` makes predictions, it automatically computes the value of the interaction `wc×age` to equal 30, as it should. Likewise, if your model includes the term `c.age##c.age`, specifying `margins`, `at(age=30)` makes predictions with age held at 30 and age-squared held at 900. This powerful feature of factor-variable notation greatly simplifies the way in which you can specify and interpret models with interactions and polynomials.

Making multiple predictions

Making multiple predictions with a single `margins` command is critical if you want to test hypotheses about those predictions, such as whether the probability of voting Republican is the same for men and for women. In this section, we consider a variety of ways to make multiple predictions.

`margins` allows you to compute multiple predictions with a single `at()` specification. For example, here we make predictions at two values of `wc`, with other variables held at their means:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 age i.wc i.hc lwg inc, nolog
(output omitted)
. margins, at(wc=0 wc=1) atmeans
```

Adjusted predictions		Number of obs	=	753
Model VCE	: OIM			
Expression	: Pr(lfp), predict()			
1._at	: k5	=	.2377158	(mean)
	: k618	=	1.353254	(mean)
	: age	=	42.53785	(mean)
	: wc	=	0	
	: 0.hc	=	.6082337	(mean)
	: 1.hc	=	.3917663	(mean)
	: lwg	=	1.097115	(mean)
	: inc	=	20.12897	(mean)
2._at	: k5	=	.2377158	(mean)
	: k618	=	1.353254	(mean)
	: age	=	42.53785	(mean)
	: wc	=	1	
	: 0.hc	=	.6082337	(mean)
	: 1.hc	=	.3917663	(mean)
	: lwg	=	1.097115	(mean)
	: inc	=	20.12897	(mean)

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_at						
1	.5215977	.0247391	21.08	0.000	.4731099	.5700855
2	.7096569	.0391445	18.13	0.000	.6329352	.7863786

The legend labeled 1._at is for `wc=0` with other variables held at the mean. The second legend, labeled 2._at, is for `wc=1`.

There are two other ways to specify these two predictions with `at()` in a single `margins` command. We could include two `at()` options in the same `margins` command:

```
margins, at(wc=0) at(wc=1) atmeans
```

Or we could use a *numlist*, Stata's name for a list of numerical values:

```
margins, at(wc=(0 1)) atmeans
```

In this specification, (0 1) is the *numlist*, which must be enclosed in parentheses. For example, `at(wc=0 1)` will generate an error.

The *atspec* can use any Stata *numlist* to specify multiple predictions (see `help numlist` for more details). One of the most useful forms allows us to specify every *#*th value over a range of values of an independent variable. For instance, say that we want predictions at every 10 years of age from 30 to 60:

```
. margins, at(age=(30(10)60)) atmeans
Adjusted predictions                                Number of obs   =       753
Model VCE      : OIM
Expression     : Pr(lfp), predict()
1._at          : k5              =       .2377158 (mean)
                  k618           =       1.353254 (mean)
                  age             =          30
                  0.wc            =       .7184595 (mean)
(output omitted)
4._at          : k5              =       .2377158 (mean)
                  k618           =       1.353254 (mean)
                  age             =          60
                  0.wc            =       .7184595 (mean)
(output omitted)
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_at						
1	.7506321	.0345282	21.74	0.000	.6829582	.818306
2	.61616	.0210803	29.23	0.000	.5748434	.6574766
3	.4612219	.0299687	15.39	0.000	.4024843	.5199594
4	.3134304	.0499253	6.28	0.000	.2155786	.4112823

If we specify *numlists* for multiple independent variables, we get predictions for all combinations of those variables. For example, to make predictions at every 10 years of age when *wc* = 0 and when *wc* = 1, holding other variables to their means, type

```
. logit lfp k5 k618 age i.wc i.hc lwg inc, nolog
(output omitted)
. margins, at(age=(30(10)60) wc=(0 1)) atmeans
Adjusted predictions      Number of obs   =      753
Model VCE      : OIM
Expression     : Pr(lfp), predict()

1._at          : k5          =      .2377158 (mean)
                k618        =      1.353254 (mean)
                age          =           30
                wc           =           0
                0.hc         =      .6082337 (mean)
                1.hc         =      .3917663 (mean)
                lwg          =      1.097115 (mean)
                inc          =      20.12897 (mean)

(output omitted)

8._at          : k5          =      .2377158 (mean)
                k618        =      1.353254 (mean)
                age          =           60
                wc           =           1
                0.hc         =      .6082337 (mean)
                1.hc         =      .3917663 (mean)
                lwg          =      1.097115 (mean)
                inc          =      20.12897 (mean)
```

	Delta-method					[95% Conf. Interval]
	Margin	Std. Err.	z	P> z		
_at						
1	.705724	.0396383	17.80	0.000	.6280344	.7834136
2	.8431665	.0339447	24.84	0.000	.7766361	.909697
3	.5611919	.0259052	21.66	0.000	.5104186	.6119651
4	.7414032	.0373631	19.84	0.000	.6681729	.8146334
5	.4054747	.0326861	12.41	0.000	.341411	.4695383
6	.604576	.0494255	12.23	0.000	.5077038	.7014483
7	.2667039	.0472562	5.64	0.000	.1740835	.3593243
8	.4491423	.0700628	6.41	0.000	.3118217	.586463

Eight predictions are made for the four values of *age* by two values of *hc*. In the *atlegend*, the values of the independent variables for each prediction are labeled as #._at, which correspond to the prediction numbers listed under *_at*.

When many predictions are calculated, the *atlegend* can take hundreds of lines. The *noatlegend* option suppresses the listing; however, although the output is shorter, it is easy to lose track of which prediction corresponds to which values of the independent variables. To address this issue, our *mlistat* command lists the *atlegend* in a more compact form:


```
. logit lfp k5 k618 age i.wc i.hc lwg inc, nolog
(output omitted)
. margins, at(age=(30(10)60) wc=(0 1)) atmeans noatlegend
Adjusted predictions                                Number of obs   =       753
Model VCE      : OIM
Expression     : Pr(lfp), predict()
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_at						
1	.705724	.0396383	17.80	0.000	.6280344	.7834136
2	.8431665	.0339447	24.84	0.000	.7766361	.909697
3	.5611919	.0259052	21.66	0.000	.5104186	.6119651
4	.7414032	.0373631	19.84	0.000	.6681729	.8146334
5	.4054747	.0326861	12.41	0.000	.341411	.4695383
6	.604576	.0494255	12.23	0.000	.5077038	.7014483
7	.2667039	.0472562	5.64	0.000	.1740835	.3593243
8	.4491423	.0700628	6.41	0.000	.3118217	.586463

```
. mlistat
at() values held constant
```

k5	k618	1. hc	lwg	inc
.238	1.35	.392	1.1	20.1

```
at() values vary
```

_at	age	wc
1	30	0
2	30	1
3	40	0
4	40	1
5	50	0
6	50	1
7	60	0
8	60	1

`mlistat` divides the independent variables into those that are constant, which are listed only once, and those that vary across predictions. If values of a variable vary, `mlistat` lists their values along with the prediction number in the `_at` column. If you do not want the output to be divided in this way, you can specify the `nosplit` option to list all values for all variables. The `noconstant` option prints only variables whose values vary.

Notice the order in which the values of `age` and `wc` vary: starting at `age=30`, the values of `wc` change from 0 to 1; then `age` changes to 40 and the values of `wc` vary; and so on. It is useful to understand what determines this order, because you might find it more useful or effective to examine the predictions arranged by `age` for a given level of `wc`, rather than the changes in `wc` for a given value of `age`. The order is determined by the order in which the independent variables appear in the variable list for the fitted model, and not by the order of variables within `at()`. Because our model was specified as `logit`

lfp k5 k618 age i.wc i.hc lwg inc, the variable wc varies first in the predictions because wc appears later in the model. If we reran the model as `logit lfp k5 k618 i.wc age i.hc lwg inc`, predictions would be in the order of wc=0 for ages 30, 40, 50, and 60, followed by predictions for wc=1 by age.

If you do not want to respecify your model to change the order of predictions (which we rarely want to do), you can use multiple `at()` specifications in the same `margins` command to control the order in which predictions are made. For example,

```
. logit lfp k5 k618 age i.wc i.hc lwg inc, nolog
(output omitted)
. margins, at(wc=0 age=(30(10)60)) at(wc=1 age=(30(10)60)) atmeans noatlegend
(output omitted)
. mlistat, noconstant
at() values vary
```

_at	age	wc
1	30	0
2	40	0
3	50	0
4	60	0
5	30	1
6	40	1
7	50	1
8	60	1

We mention this because it is important to produce predictions that make it as easy as possible to interpret results. This is considered further when we discuss the `mtable` command below.

Predictions for groups defined by levels of categorical variables

In the data we have been using, `wc` is a binary factor variable. Earlier, we showed how to get predictions for both values of `wc` by specifying a `numlist` with the `at()` option:

```
. logit lfp k5 age i.wc i.hc, nolog
(output omitted)
. margins, at(wc=(0 1)) atmeans
Adjusted predictions      Number of obs   =      753
Model VCE      : OIM
Expression      : Pr(lfp), predict()
1._at           : k5              =      .2377158 (mean)
                  age              =      42.53785 (mean)
                  wc                =              0
                  0.hc              =      .6082337 (mean)
                  1.hc              =      .3917663 (mean)
2._at           : k5              =      .2377158 (mean)
                  age              =      42.53785 (mean)
                  wc                =              1
                  0.hc              =      .6082337 (mean)
                  1.hc              =      .3917663 (mean)
```


	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_at						
1	.511531	.0237801	21.51	0.000	.464923	.5581391
2	.7214299	.0359569	20.06	0.000	.6509557	.7919041

When factor-variable notation is used for a categorical variable, the same result can be obtained by including the variables in the *varlist* for *margins*:

```
. margins wc, atmeans
Adjusted predictions
Model VCE      : OIM
Expression     : Pr(1fp), predict()
at
  k5            = .2377158 (mean)
  age           = 42.53785 (mean)
  0.wc          = .7184595 (mean)
  1.wc          = .2815405 (mean)
  0.hc          = .6082337 (mean)
  1.hc          = .3917663 (mean)
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
wc						
no	.511531	.0237801	21.51	0.000	.464923	.5581391
college	.7214299	.0359569	20.06	0.000	.6509557	.7919041

If multiple factor variables are specified in *varlist*, then *margins* computes all combinations, just as it does when a *numlist* specifies multiple variables within *at()*.

With the *at()* specification, we can use combinations of continuous variables and factor variables, whereas only factor variables can be included in the *varlist*. For instance, earlier we computed predictions over *age* by using *at(age=30(10)60)*, but typing *margins age* produces an error because *age* is not a factor variable. The *atlegend* is also more compact, and perhaps more confusing, when a *varlist* is used because it shows the means for the factor variables *0.wc* and *1.wc* even though the predictions are made with *wc=0* and *wc=1*, not at their means.

We can also make predictions at different levels of a categorical variable with the *over()* option. Like many Stata commands, *margins* supports the *over()* option to obtain separate estimates for different groups, where the variable defining the groups does not need to be in the model. There is a subtle but important difference in how *over()* makes group predictions compared with the methods considered earlier. An example is the easiest way to understand how *over()* works. For the logit model fit earlier, we make predictions over the binary variable *wc*:


```
. margins, over(wc) atmeans
```

```
Adjusted predictions      Number of obs   =       753
Model VCE      : OIM
Expression     : Pr(lfp), predict()
over           : wc
at            : 0.wc
               k5      =    .2014787 (mean)
               age     =    42.85952 (mean)
               wc      =         0
               0.hc    =    .7707948 (mean)
               1.hc    =    .2292052 (mean)
1.wc
               k5      =    .3301887 (mean)
               age     =    41.71698 (mean)
               wc      =         1
               0.hc    =    .1933962 (mean)
               1.hc    =    .8066038 (mean)
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
wc						
no	.5246101	.0224108	23.41	0.000	.4806858	.5685343
college	.6937858	.0336666	20.61	0.000	.6278004	.7597712

These results are not the same as those from `margins wc`, `atmeans` or `margins, at(wc=(0 1)) atmeans` shown above. When `over()` is used with the `atmeans` option, `margins` calculates the mean of variables within each group. You can see this in the different means listed above in the *atlegends* for `0.wc` and `1.wc`. Predictions for `wc = 0` are computed with other variables held at the mean for the subsample defined by `if wc==0`. Similarly, means for `wc = 1` are computed with `if wc==1`.

To see this, we run `margins` using an `if` condition. When an `if` or `in` condition is used with `margins`, the sample is restricted to those cases when computing means, medians, and other values for the `at()` variables. Accordingly, we can obtain identical results to those obtained using `over(wc)` by restricting the sample with an `if` condition. First, we use `if wc==0` to select observations:

```
. margins if wc==0, atmeans
```

```
Adjusted predictions      Number of obs   =       541
Model VCE      : OIM
Expression     : Pr(lfp), predict()
at            : k5      =    .2014787 (mean)
               age     =    42.85952 (mean)
               wc      =         0
               0.hc    =    .7707948 (mean)
               1.hc    =    .2292052 (mean)
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_cons	.5246101	.0224108	23.41	0.000	.4806858	.5685343

The results match those for `wc = 0` when `over(wc)` was used. Similarly with `if wc==1`:

```
. margins if wc==1, atmeans
Adjusted predictions          Number of obs   =       212
Model VCE      : OIM
Expression    : Pr(lfp), predict()
at
   k5          =    .3301887 (mean)
   age         =   41.71698 (mean)
   wc          =         1
   0.hc        =   .1933962 (mean)
   1.hc        =   .8066038 (mean)
```

	Delta-method			P> z	[95% Conf. Interval]	
	Margin	Std. Err.	z			
_cons	.6937858	.0336666	20.61	0.000	.6278004	.7597712

These results match those for `1.wc` when `over(wc)` was used.

4.4.3 (Advanced) Nondefault predictions using margins

Although the section heading seems esoteric, this is an important topic. The default predictions computed by `margins` or the `m*` commands are often the predictions you want, which is why they are the default. But you might want to predict some other quantity. Using the options described in this section, you can predict arbitrarily complex functions of any quantity computed by `predict`. Several useful applications of this powerful feature of `margins` are illustrated in later chapters.

By default, `margins` predicts whatever `predict` would predict by default for the last estimation command. For instance, the default prediction for `regress` is the predicted value $E(y|x)$, whereas for `logit` the default prediction is $\Pr(y = 1|x)$. For most estimation commands, you can predict other quantities by adding an option to `predict`. For example, after `logit`, the command `predict myxb, xb` generates the variable `myxb` with the linear combination of the x 's. To determine the default prediction and what other types of predictions are available for a given estimation command, type `help estimation-command postestimation` (for example, `help logit postestimation`). The `margins` command can estimate any of the quantities computed by `predict`, as well as arbitrarily complex functions of these quantities with the `predict()` and `expression()` options.

The `predict()` option

With the `predict(statistic)` option, the `margins` command makes predictions for any *statistic* that can be computed by `predict`. For example, in the ordered logit model


```

. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. margins, expression(1-predict(pr))
Predictive margins                                Number of obs   =       753
Model VCE      : OIM
Expression     : 1-predict(pr)

```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_cons	.4316069	.0166014	26.00	0.000	.3990688	.464145

The `expression()` option is incredibly powerful, allowing you to routinely test things that would otherwise be difficult. To test whether the expected probability is 0.5 after `logit`, use `margins, expression(predict(pr)-.5)`. In this expression, `predict(pr)` computes the probability that $y = 1$. By subtracting 0.5, we are computing deviations from 0.5. We can use the z statistic from `margins` to test whether the average deviation is 0, which is equivalent to testing whether the average probability is 0.5. As another example, after `ologit` we can test whether the probability of a respondent identifying as lower class ($y = 1$) equals the probability of identifying as upper class ($y = 4$) by using

```
margins, expression( predict(outcome(1)) - predict(outcome(4)) )
```

In chapter 6, we show how this feature can be particularly handy when working with independent variables that have power terms or interactions with other independent variables in the model.

4.4.4 Tables of predictions using *mtable*

`mtable` makes tables from the predictions computed by `margins`. You do not need to run `margins` because `mtable` does this for you, using most of the options for `margins` that we just considered. In addition, `mtable` has options to customize how the results appear by adding labels, selecting statistics, and combining results from multiple `mtable` commands. There are, however, some features in `margins` that will not work with `mtable`. Most notably, perhaps, `margins` allows a *varlist* with factor variables, but `mtable` does not. But as we showed on page 151, results that can be computed with a *varlist* can be computed using `at()`, so this does not limit what you can do with `mtable`.

To explain how `mtable` works, we start by creating a table of predicted probabilities that vary by `wc` and `hc` from a `logit` model. We will talk at length about how to interpret these predictions in chapter 6.


```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. mtable, at(wc=(0 1) hc=(0 1)) atmeans
Expression: Pr(lfp), predict()
```

	wc	hc	Pr(y)
1	0	0	0.509
2	0	1	0.543
3	1	0	0.697
4	1	1	0.725

Specified values of covariates

	k5	k618	2. agecat	3. agecat	lwg	inc
Current	.238	1.35	.385	.219	1.1	20.1

In the header, `Expression` echoes the description that `margins` uses to describe the predictions it is making. The column `Pr(y)` contains predicted probabilities that `lfp` is 1. The first row of the prediction table, numbered 1, shows that the probability of being in the labor force is 0.509 for a woman who did not go to college (`wc=0`) and whose husband did not go to college (`hc=0`), holding other variables at their means as specified with the `atmeans` option. Rows 2, 3, and 4 show predictions for other combinations of `hc` and `wc`. Values of the independent variables that are held constant are displayed below the predictions.

To convince you (we hope) of the advantages of `mtable`, let's look at the corresponding output from `margins`. We show all the output because if you use `noatlegend`, you risk not knowing which predictions correspond to which values of the variables that vary.

```
. margins, at(wc=(0 1) hc=(0 1)) atmeans
Adjusted predictions
Model VCE      : OIM
Expression     : Pr(lfp), predict()
Number of obs   =      753
```

1._at	: k5	=	.2377158 (mean)
	: k618	=	1.353254 (mean)
	: 1.agecat	=	.3957503 (mean)
	: 2.agecat	=	.3851262 (mean)
	: 3.agecat	=	.2191235 (mean)
	: wc	=	0
	: hc	=	0
	: lwg	=	1.097115 (mean)
	: inc	=	20.12897 (mean)
2._at	: k5	=	.2377158 (mean)
	: k618	=	1.353254 (mean)
	: 1.agecat	=	.3957503 (mean)
	: 2.agecat	=	.3851262 (mean)
	: 3.agecat	=	.2191235 (mean)
	: wc	=	0
	: hc	=	1


```

          lwg      = 1.097115 (mean)
          inc      = 20.12897 (mean)
3._at      : k5      = .2377158 (mean)
              k618    = 1.353254 (mean)
              1.agecat = .3957503 (mean)
              2.agecat = .3851262 (mean)
              3.agecat = .2191235 (mean)
              wc       = 1
              hc       = 0
              lwg      = 1.097115 (mean)
              inc      = 20.12897 (mean)
4._at      : k5      = .2377158 (mean)
              k618    = 1.353254 (mean)
              1.agecat = .3957503 (mean)
              2.agecat = .3851262 (mean)
              3.agecat = .2191235 (mean)
              wc       = 1
              hc       = 1
              lwg      = 1.097115 (mean)
              inc      = 20.12897 (mean)

```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_at						
1	.5090529	.0274599	18.54	0.000	.4552326	.5628733
2	.5429204	.0444615	12.21	0.000	.4557775	.6300633
3	.6971851	.0489817	14.23	0.000	.6011828	.7931874
4	.7250834	.0364056	19.92	0.000	.6537297	.796437

The margins output has additional information about the predictions, such as the confidence interval, that was missing from the `mtable` output.

We can include the confidence interval in the `mtable` output by adding the option `statistics(ci)`. At the same time, we show how to customize the label for predictions by using `estname()`:

```
. mtable, at(wc=(0 1) hc=(0 1)) atmeans estname(Pr_LFP) statistics(ci)
```

```
Expression: Pr(lfp), predict()
```

	wc	hc	Pr_LFP	l1	u1
1	0	0	0.509	0.455	0.563
2	0	1	0.543	0.456	0.630
3	1	0	0.697	0.601	0.793
4	1	1	0.725	0.654	0.796

```
Specified values of covariates
```

	k5	k618	2. agecat	3. agecat	lwg	inc
Current	.238	1.35	.385	.219	1.1	20.1

The `statistics(statlist)` option allows you to add other statistics as well. The potential elements of `statlist` are shown in the following table.

Statistics	Description
<code>est</code>	Estimate
<code>ci</code>	Confidence intervals along with the estimate
<code>ll, ul</code>	Lower, upper limit of confidence interval
<code>se</code>	Standard error of the estimate
<code>z</code>	z statistic for test that estimate is 0
<code>pvalue</code>	p -value for test that estimate is 0
<code>all</code>	All the above statistics

By default, `mtable` includes columns with the values of the `at()` variables that are changing. If you do not want these columns displayed, specify `atvars(_none)`. You can use the `atvars(varlist)` option to select which variables will appear, even if their values are not changing. This is useful when building tables (discussed soon) or when you want your table to show the level of a variable that is not varying across the predictions. In our example, `wc` varies but `k5` and `k618` do not. To include them in the table, we add the option `atvars(k5 k618 wc)`. We also use the `brief` option so that the table of values for covariates is not shown:

```
. mtable, at(k5=0 k618=0 wc=(0 1)) atmeans atvars(k5 k618 wc) brief
Expression: Pr(lfp), predict()
```

	k5	k618	1. wc	Pr(y)
1	0	0	0	0.625
2	0	0	1	0.787

mtable with categorical and count outcomes

With categorical and count outcomes, a separate `margins` command must be run for each value of the outcome variable. For example, to compute predictions for each outcome category in an ordinal logit, you would need a series of commands such as `margins, predict(outcome(1)) atmeans`, then `margins, predict(outcome(2)) atmeans`, and so on. In contrast, `mtable` automatically calculates predicted probabilities for all categories. Indeed, automatically computing predictions for multiple outcomes and combining the predictions into a single table is what initially motivated the creation of `mtable`.

In one of our running examples in chapter 7, the outcome is subjective class identification, with categories ranging from 1 for lower class to 4 for upper class. To examine how attitudes are related to a respondent's race and gender, we compute the following

table, where the `dec(2)` option indicates that 2 decimal places should be used to display the estimates:

```
. use gssclass4, replace
(gssclass4.dta | GSS Subjective Class Identification | 2013-11-20)
. ologit class i.fem i.white i.year i.ed age inc, nolog
(output omitted)
. mtable, at(fem=(0 1) white=(0 1)) atmeans stat(ci) dec(2)
Expression: Pr(class), predict(outcome())
```

	female	white	lower	working	middle	upper
Pr(y)	0	0	0.06	0.52	0.41	0.01
ll	0	0	0.05	0.49	0.38	0.01
ul	0	0	0.07	0.54	0.44	0.02
Pr(y)	0	1	0.05	0.46	0.47	0.02
ll	0	1	0.04	0.44	0.45	0.01
ul	0	1	0.05	0.48	0.49	0.02
Pr(y)	1	0	0.06	0.51	0.42	0.01
ll	1	0	0.05	0.48	0.38	0.01
ul	1	0	0.07	0.54	0.45	0.02
Pr(y)	1	1	0.05	0.46	0.48	0.02
ll	1	1	0.04	0.44	0.46	0.01
ul	1	1	0.05	0.48	0.50	0.02

Specified values of covariates

	2. year	3. year	2. educ	3. educ	age	income
Current	.45	.31	.58	.24	45	68

Holding other variables at their means, the predicted probability of identifying as working class is 0.52 for nonwhite men (`fem=0`, `white=0`). With categorical outcomes, additional statistics are placed below the estimates, in this case showing the lower and upper bounds for the confidence interval.

By default, all outcome categories are included in the table. The options `pr(numlist)` and `outcome(numlist)` let you select which outcomes to include in the table. For estimation commands that support the `outcome()` option in `predict` (for example, `ologit`), `mtable` uses the `outcome()` option to select which predictions to present. For commands such as `logit` and `poisson` that support the `pr` option with `predict`, `mtable` uses `pr()` to select which outcomes to present. For example, if we want to display only results for the 1=lower class and 4=upper class categories, we type

```
. mtable, at(fem=(0 1) white=(0 1)) outcome(1 4) atmeans brief
Expression: Pr(class), predict(outcome())
```

	female	white	lower	upper
1	0	0	0.061	0.014
2	0	1	0.048	0.018
3	1	0	0.060	0.014
4	1	1	0.047	0.018

where we suppress the values of the covariates with `brief`.

For count models discussed in chapter 9, the default prediction for `mtable` is the rate because it is the default for `predict` in count models. To display predicted probabilities for a specific count—say, 0—we would use the option `pr(0)`. To compute predicted probabilities for counts from 0 to 5, we would use `pr(0/5)`.

(Advanced) Combining and formatting tables using `mtable`

We mark this section as advanced because it does not consider new ways of making predictions using `mtable`, but instead considers how to create tables that combine predictions from running multiple `mtable` commands. If you are only making a few predictions, the time it takes to learn these features might not be worth it. But if you often create tables of predictions, these features will save you time, make it easier to see key results, and prevent the inevitable errors that occur when constructing tables by hand.

`mtable` allows you to combine results from multiple `mtable` commands. The best way to understand how this works is with an example. Suppose that we want to compare the average predicted probability of labor force participation with the predicted probability holding all variables at their mean. We begin by fitting the model:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
```

Using the results from this model, we next estimate the average probability of labor force participation for values of `k5` from 0 to 3. Because our table is going to include multiple predictions, we add labels to identify each set of predictions. The option `coleqnm()` adds a header row to our predictions. The name for this option might seem odd, but it reflects that `mtable` saves results as matrices that refer to this header as the “column equation name”. We use the `coleqnm(1st)` option to label our first set of predictions:

```
. mtable, at(k5=(0 1 2 3)) coleqnm(1st)
Expression: Pr(lfp), predict()
```

	1st	
	k5	Pr(y)
1	0	0.637
2	1	0.342
3	2	0.129
4	3	0.038

Specified values where .n indicates no values specified with `at()`

	No at()
Current	.n

Next, we compute predictions at the means by adding the `atmeans` option, using the `right` option to append the new predictions to the right of those above. To simplify our example, we exclude the `atlegend` with the `brief` option:

```
. mtable, at(k5=(0 1 2 3)) atmeans right brief coleqnm(2nd)
Expression: Pr(lfp), predict()
```

	1st	2nd			
	k5	Pr(y)	k5	Pr(y)	
1	0	0.637	0	0.656	
2	1	0.342	1	0.322	
3	2	0.129	2	0.105	
4	3	0.038	3	0.028	

The results on the left labeled **1st** are from the first time we ran `mtable` to compute the average predicted probability. The next two columns, labeled **2nd**, are predictions at the mean.

Soon we will show how to make the output more effective by removing the repetition of the `k5` column and using better labels. First, however, we need to explain how the levels of covariates are displayed when combining tables. Here is the output we would obtain if we had not used the `brief` option:

```
. mtable, at(k5=(0 1 2 3)) atmeans right coleqnm(2nd)
Expression: Pr(lfp), predict()
```

	1st	2nd			
	k5	Pr(y)	k5	Pr(y)	
1	0	0.637	0	0.656	
2	1	0.342	1	0.322	
3	2	0.129	2	0.105	
4	3	0.038	3	0.028	

Specified values where .n indicates no values specified with `at()`

	No at()	k618	2. agecat	3. agecat	1. wc	1. hc
Set 1	.n
Current	.	1.35	.385	.219	.282	.392
	lwg	inc				
Set 1	.	.				
Current	1.1	20.1				

The row labeled **Set 1** contains values from the first use of `mtable` whose predictions are labeled **1st**. The `.n` in the column labeled **No at()** indicates that the predictions were made without an `at()` specification; `.n` stands for “no covariates specified”. The second row, labeled **Current**, lists the mean values of each variable from the most recent (that is, current) use of `mtable`; these correspond to the predictions in the columns labeled **2nd**.

We can make the same table more clear by using additional options. First, there is no reason for the values of `k5` to appear twice. Specifying `atvars(_none)` in the second call of `mtable` removes this redundancy. Second, the two columns of predicted probabilities should have different labels, which is accomplished with the `estname()` option. We will call them `asobserved` and `atmeans`:

```
. quietly mtable, at(k5=(0 1 2 3)) estname(asobserved)
. mtable, at(k5=(0 1 2 3)) atmeans atvars(_none) estname(atmeans) right brief
Expression: Pr(lfp), predict()
```

	k5	asobserved	atmeans
1	0	0.637	0.656
2	1	0.342	0.322
3	2	0.129	0.105
4	3	0.038	0.028

Next, we add a title to the table with the option `title(Predicted probability of labor force participation)`. Finally, the numbers in the left column are not needed because the content of the rows is clear from the levels of `k5`. We can remove these with the `norownum` option. Combining these, we get a table that is close to what we might include in a paper:

Predicted probability of labor force participation

Expression: Pr(lfp), predict()

k5	asobserved	atmeans
0	0.637	0.656
1	0.342	0.322
2	0.129	0.105
3	0.038	0.028

We use `mtable` often in the later chapters, where we take advantage of its many formatting features. Type `help mtable` to see them all.

4.5 Marginal effects: Changes in predictions

Marginal effects are estimates of the change in an outcome for a change in one independent variable, holding all other variables constant. Here we provide an overview of the commands and basic concepts for computing marginal effects. A detailed discussion of marginal effects, along with substantive applications of alternative measures of change, is given in later chapters, especially chapter 6. We begin by discussing `margins`, which computes marginal effects with the `dydx()` option, and we then show how `mtable` can do the same thing. Because marginal effects are such a useful summary of effects in nonlinear models, we created `mchange` to easily compute many types of changes and present them in a compact table.

4.5.1 Marginal effects using margins

`margins` can calculate the change in a predicted quantity as an independent variable changes, holding other variables constant. The prediction can be anything that `margins` can estimate. The variables for which changes are calculated are specified using the `dydx(varlist)` option, where `dydx(*)` indicates that changes for all independent variables are to be computed. For example,

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 i.agecat i.wc inc, nolog
(output omitted)
. margins, dydx(*)
Average marginal effects      Number of obs   =      753
Model VCE      : OIM
Expression     : Pr(lfp), predict()
dy/dx w.r.t.   : k5 2.agecat 3.agecat i.wc inc
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	dy/dx	Std. Err.				
k5	-.2886234	.034555	-8.35	0.000	-.35635	-.2208968
agecat						
40-49	-.1081118	.0397652	-2.72	0.007	-.1860501	-.0301735
50+	-.2424494	.0457377	-5.30	0.000	-.3320937	-.152805
wc						
college	.2217304	.0359796	6.16	0.000	.1512116	.2922492
inc	-.006773	.0015749	-4.30	0.000	-.0098597	-.0036863

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

The amount of change in a regressor that is used to calculate the change in the prediction depends on whether the variable is a continuous or a factor variable, where Stata assumes variables are continuous unless specified as factor variables with the `i.` notation. In our example, `k5` and `inc` are continuous while `agecat` and `wc` are factor variables. For a continuous variable, `margins` estimates the marginal change, which is the partial derivative or instantaneous rate of change in the estimated quantity with respect to a given variable, holding other variables constant. For factor variables, `margins` calculates the discrete change, which is the difference in the prediction when the factor variable is 1 compared with the prediction when the variable is 0. For the binary variable `wc`, this is the change in the probability of being in the labor force if the wife attended college compared with if she did not attend college. For multiple-category factor variables, the change is from the base category to the value listed in column `dy/dx`. For `i.agecat` in this example, the row labeled 40-49 is the change in the probability for a change in `agecat` from the excluded base category 30-39 to the category 40-49.

It bears repeating that `margins` only calculates the discrete change for variables specified with the `i.` factor-variable notation. For example (using underlining to high-

light the differences between the two commands), although `logit lfp k5 i.agecat wc inc` and `logit lfp k5 i.agecat i.wc inc` yield the same estimates of the regression coefficients, the values of `dydx()` computed by `margins` will differ. In the first specification, `wc` is not a factor variable, so `margins` computes the partial derivative with respect to `wc`; in the second specification, `i.wc` is a factor variable, so `margins` computes the discrete change. Almost certainly in this context, you want the discrete change, and so factor-variable notation must be used when fitting the model.

4.5.2 Marginal effects using `mtable`

Showing how `mtable` can compute the same results as `margins` provides an opportunity to illustrate the mechanics of how discrete changes are computed and to extend our discussion of how `mtable` can display different estimates in a single table. We use a pair of `mtable` commands to compute predicted probabilities, first when `wc` is 1 and then when `wc` is 0. Using two `mtable` commands rather than a single command with `at(wc=(0 1))` allows us to have different row labels for each prediction by adding the `rowname()` option. The option `below` indicates that the results from the second `mtable` command should be stacked below those from the first `mtable`:

```
. quietly mtable, at(wc=1) rowname(wc=1) statistics(ci) estname(Pr_LFP)
. mtable, at(wc=0) rowname(wc=0) statistics(ci) estname(Pr_LFP) below
Expression: Pr(lfp), predict()
```

	Pr_LFP	ll	ul
wc=1	0.728	0.671	0.784
wc=0	0.506	0.466	0.546

Specified values of covariates

	wc
Set 1	1
Current	0

The first row contains the average predicted probability of labor force participation under the assumption that all women went to college, while the second row contains predictions assuming that none of the women went to college. Next, we compute the discrete change by using the `dydx(wc)` option:

```
. mtable, dydx(wc) rowname(wc=1 - wc=0) statistics(ci) estname(Pr_LFP)
> below brief
Expression: Pr(lfp), predict()
```

	Pr LFP	ll	ul
wc=1	0.728	0.671	0.784
wc=0	0.506	0.466	0.546
wc=1 - wc=0	0.222	0.151	0.292

The results match those for `wc` from `margins` on page 163.

4.5.3 Posting predictions and using `margins`

Computing change is also an ideal venue for introducing the idea of posting estimates. To explain this powerful feature of `margins`, we need to review how estimation commands work. After a regression model is fit, Stata saves the results in memory as what are called `ereturns`. The coefficient estimates are saved in the matrix `e(b)` and the variance-covariance of the estimates in `e(V)`. Commands may subsequently use `ereturns` to compute additional quantities. For example, the `test` command uses `e(b)` and `e(V)` to compute Wald tests of linear hypotheses, and `lincom` uses these matrices to estimate linear combinations of the estimates. We could also test nonlinear hypotheses with `testnl` or compute nonlinear functions of estimates with `nlcom`.

Just like regression estimation commands, `margins` computes estimates and their variance-covariance matrix. By default, these are saved in the return matrices `r(b)` and `r(V)` so that `margins` does not overwrite `e(b)` and `e(V)` from the regression model. Because the results of the regression model are not disturbed, we can run multiple `margins` commands without refitting the regression model. The `post` option for `margins` replaces the matrices `e(b)` and `e(V)` from the regression model with the estimates from `margins`. Once this is done, `test` can be used to test linear hypotheses about the predictions from `margins`, and `lincom` can be used to estimate linear combinations of the predictions. Adding the `post` option to `mtable` does the same thing.³

To illustrate posting, we compute the discrete change for `wc` from the last example. We start by saving the estimation results from `logit` so that we can restore them after we finish analyzing the estimates with `mtable`.

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 i.agecat i.wc inc, nolog
(output omitted)
. estimates store model1
```

With `mtable`, we compute predicted probabilities at two levels of `wc` and use the `post` option to save the estimated predictions and their covariance matrix to `e(b)` and `e(V)`:

```
. mtable, at(wc=(1 0)) post
Expression: Pr(lfp), predict()


|   | wc | Pr(y) |
|---|----|-------|
| 1 | 1  | 0.728 |
| 2 | 0  | 0.506 |


Specified values where .n indicates no values specified with at()


|         | No at() |
|---------|---------|
| Current | .n      |


```

3. If you are using `mtable` with a model with multiple outcomes (such as `mlogit`), then `mtable` runs `margins` once for each outcome. In this case, `mtable, post` posts the `margins` estimates for the last outcome.

To see the posted predictions, we list `e(b)`:

```
. matlist e(b)
```

	1. _at	2. _at
y1	.7276954	.505965

Next, we estimate the differences between these predictions by using `lincom`. The `lincom` command requires us to specify the difference with the symbolic names of the estimates, which are shown as the column names of `e(b)`:

```
. lincom _b[1._at] - _b[2._at]
(1) 1bn._at - 2._at = 0
```

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
(1)	.2217304	.0359796	6.16	0.000	.1512116 .2922492

The estimated discrete change of 0.2217 matches the earlier results from `dydx(wc)`.

Personally, we find names like `_b[1._at]` and such to be cumbersome, so we created the `mllincom` command, which allows you to refer to an estimate by its position rather than by its name. Here we compute the difference between the first and second predictions:

```
. mllincom 1 - 2
```

	lincom	pvalue	ll	ul
1	0.222	0.000	0.151	0.292

Options for `mllincom` (type `help mllincom` for details) allow you to select which statistics you want to see, add labels, and combine results from multiple `mllincom` commands.

Regardless of whether we used `mllincom` or `lincom`, the estimation results from `logit` are no longer active. To run additional `margins` or `m*` commands, or to use `test` or `lrtest` on the regression estimates, we must restore the logit estimates:

```
. estimates restore model1
(results model1 are active now)
```

In this simple example, there is no advantage to using `post` and `mllincom` because `mtable`, `dydx(wc)` is much simpler. In later chapters, though, we use `mllincom` to test a variety of more complex and useful hypotheses about marginal effects.

4.5.4 Marginal effects using `mchange`

Marginal effects are so fundamental for interpreting nonlinear models that we created `mchange` to make it simple to create compact tables containing many types of marginal effects. By default, changes are computed `asobserved`. That is, the change is computed

for each observation in the estimation sample and then averaged. These are sometimes referred to as average marginal effects. If we want the marginal effects at the mean instead, we can use the *atmeans* option, or we can set specific values of the independent variables at which changes are computed by using the *at()* option.

To see how this command differs from *margins*, *dydx(*)*, let's first consider what *mchange* provides by default:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. mchange
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

		Change	p-value
k5			
	+1	-0.281	0.000
	+SD	-0.153	0.000
	Marginal	-0.289	0.000
k618			
	+1	-0.014	0.337
	+SD	-0.018	0.337
	Marginal	-0.014	0.335
agecat			
	40-49 vs 30-39	-0.124	0.002
	50+ vs 30-39	-0.262	0.000
	50+ vs 40-49	-0.138	0.002
wc			
	college vs no	0.162	0.000
hc			
	college vs no	0.028	0.508
lwg			
	+1	0.120	0.000
	+SD	0.072	0.000
	Marginal	0.127	0.000
inc			
	+1	-0.007	0.000
	+SD	-0.086	0.000
	Marginal	-0.007	0.000
Average predictions			
		not in LF	in LF
Pr(y base)		0.432	0.568

For continuous variables (that is, those not specified as *i.varname*), the rows labeled *Marginal* contain the average marginal changes and are identical to what is computed using *margins*, *dydx(*)*. For binary factor variables, such as *i.wc*, *mchange* computes the average discrete change as the variable changes from 0 to 1. In Stata 13 and later, value labels are used to label the change if available (for example, *college vs no*); otherwise, values are used (*1 vs 0*). For categorical factor variables, such as *i.agecat*, *mchange* computes differences between all pairs of categories (referred to as contrasts).

In Stata 13 and later, these contrasts are labeled with the value labels associated with the categories. The contrasts 40-49 vs 30-39 and 50+ vs 30-39 are the same as shown in the output from `margins, dydx(*)`. The comparison 50+ vs 40-49 was computed by `mchange` with `margins, pwcompare`.

By default, for continuous variables, `mchange` also computes two types of discrete changes. The unit change, labeled +1, is the change in the prediction as a variable changes from its observed value to its observed value plus 1. The standard deviation change, labeled +SD, is the change in the prediction as a variable changes one standard deviation from its observed value.

`mchange` has many options, a few of which we discuss here. A full description of `mchange`'s functionality is provided with `help mchange`, and many examples are provided in later chapters.

The `amount(amount-types)` option specifies the amount of change for continuous variables. The following are available:

<i>amount-type</i>	Amount of change
<code>one</code>	One unit change
<code>sd</code>	Standard deviation change
<code>marginal</code>	Marginal change
<code>binary</code>	Change from 0 to 1
<code>range</code>	Change from the minimum to maximum
<code>all</code>	All the above

The default amounts are `amount(one sd marginal)`, which were described above. In Stata 11 and 12, changes of 1 or 1 standard deviation cannot be computed and will appear as .m in the results. By default, changes of 1 and 1 standard deviation are uncentered. The `centered` option requests changes that are centered; a centered unit change is the change from $x - (1/2)$ to $x + (1/2)$ rather than from x to $x + 1$.

The `amount(range)` option computes the discrete changes as x changes from its minimum to maximum, but it can also be used with the `trim(#)` option to compute the change between other percentiles. For example, we could estimate the average change in the probability of labor force participation if family income changes from the 25th percentile of income to the 75th percentile by using `trim(25)` with `amount(range)`:


```
. mchange inc, amount(range) trim(25)
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

	Change	p-value
inc		
25% to 75%	-0.084	0.000
Average predictions		
	not in LF	in LF
Pr(y base)	0.432	0.568

The `delta(#)` option computes a change of # units instead of a standard deviation change. For example, `inc` is measured in thousands of dollars. To estimate the average change in labor force participation if income increased by \$5,000, we use `delta(5)`:

```
. mchange inc, amount(sd) delta(5)
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

	Change	p-value
inc		
+delta	-0.037	0.000
Average predictions		
	not in LF	in LF
Pr(y base)	0.432	0.568

```
i: Delta equals 5.
```

The `statistics(statistics-types)` option allows you to select which statistics related to the marginal effect to display. By default, `mchange` provides the estimated change and the *p*-value from a test of the hypothesis that the effect is 0. The following statistics are available:

<i>statistics-type</i>	Statistic
<code>change</code>	Estimated change
<code>ci</code>	Estimated change and confidence interval
<code>ll, ul</code>	Lower, upper limit of confidence interval
<code>se</code>	Standard error of the estimate
<code>z</code>	<i>z</i> statistic for test that estimated change is 0
<code>pvalue</code>	<i>p</i> -value for test that estimated change is 0
<code>start</code>	Prediction at starting value of discrete change
<code>end</code>	Prediction at ending value of discrete change
<code>all</code>	All the above statistics

We can use `mchange`, `amount(all)` `statistics(all)` to compute all the statistics for all types of changes, which is a lot of information. For just one variable, here is what we get:

```
. mchange inc, amount(all) statistics(all)
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

	Change	p-value	LL	UL	z-value	Std Err
inc						
0 to 1	-0.006	0.000	-0.009	-0.004	-5.454	0.001
+1	-0.007	0.000	-0.011	-0.004	-4.419	0.002
+SD	-0.086	0.000	-0.124	-0.048	-4.404	0.019
Range	-0.593	0.000	-0.761	-0.424	-6.883	0.086
Marginal	-0.007	0.000	-0.010	-0.004	-4.427	0.002
	From	To				
inc						
0 to 1	0.706	0.700				
+1	0.568	0.561				
+SD	0.568	0.483				
Range	0.706	0.114				
Marginal	.z	.z				
Average predictions						
	not in LF	in LF				
Pr(y base)	0.432	0.568				

By selecting which statistics we want to show, `mchange` can easily replicate what took several steps with `mtable` earlier. We use a *varlist* to select variable `wc` and option `statistics(start end est pvalue)` to display the start and end values leading to the change that is shown along with its *p*-value:

```
. mchange wc, statistics(start end est pvalue)
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

	From	To	Change	p-value
wc				
college vs no	0.525	0.688	0.162	0.000
Average predictions				
	not in LF	in LF		
Pr(y base)	0.432	0.568		

By default, `mchange` computes average marginal effects (see section 6.2 for a detailed discussion). You can compute marginal effects at the mean by adding the `atmeans` option. Or you can use `at()` to compute changes at specific values of the independent variables. Finally, if you use factor-variable notation to specify interactions or polynomial terms, `mchange` will compute marginal effects by making the appropriate changes among linked variables. For example, if your model includes `c.age##c.age`,

then `mchange age` will change both age and age-squared. The `mchange` command does a lot of work behind the scenes and can take a long time to run in models such as `ologit` or `mlogit` when there are many outcome categories. For example, after running our baseline model for `ologit` with four categories and eight variables, `mchange, amount(all)` runs 40 `margins` commands, 32 `lincom` commands, and summarizes 1,123 lines of output in a 46-line table.

4.6 Plotting predictions

For continuous variables, graphs can effectively summarize effects. The Stata command `marginsplot` plots the predictions from the most recently run `margins`. Our `mgen` command can also be used to plot results from `margins`. The major difference between the commands is that `marginsplot` creates plots, while `mgen` generates variables that can be used with Stata's graphing commands. The former approach is convenient, but ultimately limited because it allows you to plot only a single outcome category from a single model in a graph.

4.6.1 Plotting predictions with marginsplot

Stata's documentation has an especially detailed discussion of what can be done with `marginsplot`. Mitchell (2012a) also provides many examples of using `marginsplot` for both linear and categorical outcome models. `marginsplot` uses results from the preceding `margins` command. For example, here we plot the predicted probabilities of labor force participation over the ages 20 to 80 for women who attended college and those who did not:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 age i.wc i.hc lwg inc, nolog
(output omitted)
. margins, at(age=(20(10)80) wc=(0 1)) atmeans
Adjusted predictions      Number of obs   =      753
Model VCE      : OIM
Expression     : Pr(lfp), predict()
1._at          : k5              =    .2377158 (mean)
                  k618           =    1.353254 (mean)
                  age            =          20
                  wc             =           0
                  0.hc           =    .6082337 (mean)
                  1.hc           =    .3917663 (mean)
                  lwg            =    1.097115 (mean)
                  inc            =    20.12897 (mean)
(output omitted)
```



```

14._at      : k5          = .2377158 (mean)
              k618        = 1.353254 (mean)
              age          =      80
              wc           =      1
              0.hc         = .6082337 (mean)
              1.hc         = .3917663 (mean)
              lwg          = 1.097115 (mean)
              inc          = 20.12897 (mean)

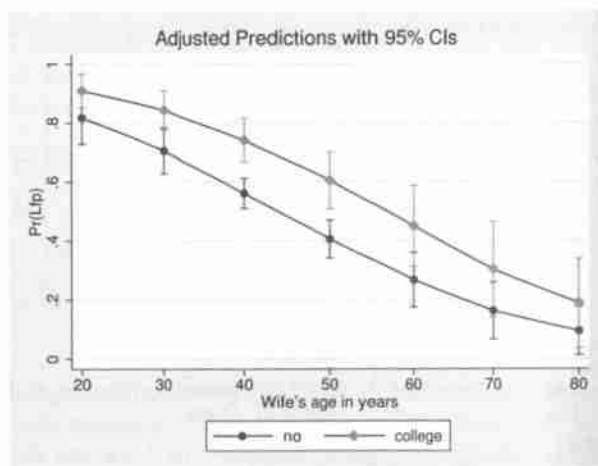
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_at						
1	.8180827	.0457432	17.88	0.000	.7284277	.9077378
2	.9097581	.0291986	31.16	0.000	.8525299	.9669863
3	.705724	.0396383	17.80	0.000	.6280344	.7834136
4	.8431665	.0339447	24.84	0.000	.7766361	.909697
5	.5611919	.0259052	21.66	0.000	.5104186	.6119651
6	.7414032	.0373631	19.84	0.000	.6681729	.8146334
7	.4054747	.0326861	12.41	0.000	.341411	.4695383
8	.604576	.0494255	12.23	0.000	.5077038	.7014483
9	.2667039	.0472562	5.64	0.000	.1740835	.3593243
10	.4491423	.0700628	6.41	0.000	.3118217	.586463
11	.1624493	.0492577	3.30	0.001	.0659059	.2589927
12	.3030444	.0818881	3.70	0.000	.1425467	.4635422
13	.0937383	.0413068	2.27	0.023	.0127785	.1746981
14	.1882307	.0768769	2.45	0.014	.0375547	.3389067

Typing `marginsplot` without any options produces the following graph:

```
. marginsplot
```

```
Variables that uniquely identify margins: age wc
```



Impressively, `marginsplot` infers that because the margins predictions differ over age and `wc`, we want to plot predictions as these variables vary. Moreover, because age is

estimated as a continuous variable and *wc* as a factor variable, it assumes that we want to make a plot in which our *x* axis is *age* and different lines represent different values of *wc*. *marginsplot* has dozens of its own options and allows you to include options for twoway graphs, such as those for axes and title. For details, type *help marginsplot*.

With *marginsplot*, you can quickly create graphs of the predictions from the last *margins* command. You often do not have to specify any options for it to create the graph you might want. Using its many options, you can customize the defaults to create publication-quality graphs. There is, however, a major limitation in what *marginsplot* can do. Graphs created by *marginsplot* can include multiple lines, such as the two lines in our example for those who went to college and those who did not. However, these plot lines must be for predictions of the same quantity computed from the same model. Among other implications, this means that *marginsplot* does not allow you to plot multiple outcomes in a single graph, which you would commonly do with ordinal or nominal outcomes. Nor can you compare predictions from two models. For example, you could not compare the predictions from a model that included *age* with one that included *age* and *age-squared*. To facilitate making such graphs, we created the *mgen* command.

4.6.2 Plotting predictions using *mgen*

The *mgen* command generates variables that can be plotted using Stata's *graph* commands. Like *mtable* and *mchange*, *mgen* runs *margins* for you and accepts most of the options that can be used with *margins*. The most important options for graphing are *at()*, which is used to specify the range of the variable on the *x* axis and the levels of other variables, and *atmeans*, if you want to hold other variables at the mean.

Here is a simple example that uses *mgen* to create a variable containing predictions as income increases from \$0 to \$100,000 in increments of \$10,000:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. mgen, at(inc=(0(10)100)) stub(A) atmeans
Predictions from: margins, at(inc=(0(10)100)) atmeans predict(pr)
Variable   Obs Unique      Mean      Min      Max Label
-----
Apr1       11      11   .3608011   .0768617   .7349035 pr(y=in LF) from margins
All1       11      11   .2708139  -.0156624   .6641427 95% lower limit
Aul1       11      11   .4507883   .1693859   .8056643 95% upper limit
Ainc       11      11       50         0       100 Family income excluding...
```



```
Specified values of covariates
```

	k5	k618	2. agecat	3. agecat	1. wc	1. hc	lwg
	.2377158	1.353254	.3851262	.2191235	.2815405	.3917663	1.097115

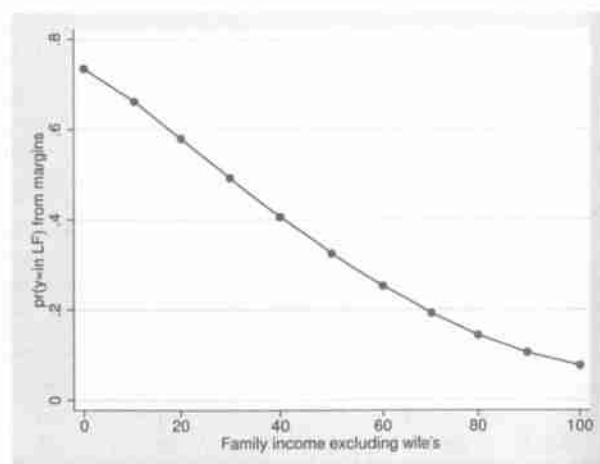
The option `stub(stubname)` provides the first letters to be used in the names of the variables that are generated. We recommend a stub that differs from the starting letters of variables in the dataset; then, afterward, the variables can be easily deleted by typing `drop stubname*`. If the variable names in your dataset are all lowercase, an uppercase stub works well for this purpose. If you do not use the `stub()` option, the default stub is an underscore, leading to variable names such as `_pr1`. If you want to overwrite existing variables, perhaps while debugging the command, you can include the option `replace`.

In our example, `mgen` generated four variables: `Apr1` with the predicted probabilities, `All1` and `Aul1` with the lower and upper bounds of the confidence interval for the prediction, and `Ainc` with values of `inc` for each prediction. The values of `Ainc` are determined by the `at()` option. The summary statistics for generated variables show that `inc` ranges from 0 to 100, with predicted probabilities ranging from 0.08 to 0.73. We can list these values:

```
. list Apr Ainc in 1/12, clean
      Apr1  Ainc
1.   .7349035    0
2.   .6613024   10
3.   .5789738   20
4.   .4920058   30
5.   .4055189   40
6.   .324523    50
7.   .2528245   60
8.   .1924536   70
9.   .1437253   80
10.  .1057196   90
11.  .0768617  100
12.  .          .
```

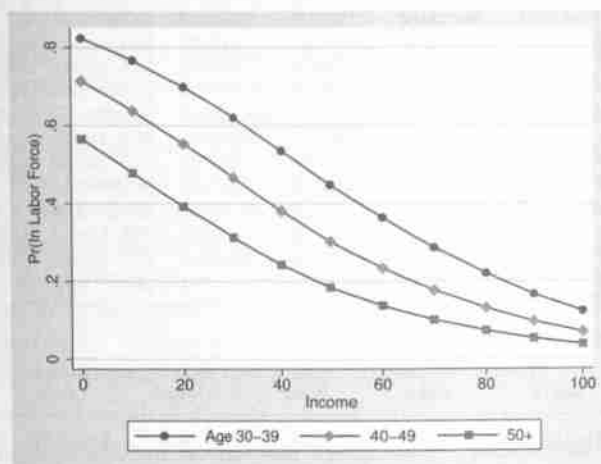
Because the predictions are saved as variables, they can be plotted with `graph`:

```
. graph twoway connected Apr Ainc
```



We can run `mgen` multiple times to generate variables with predictions at different levels of variables that are not varying. Here we use `quietly` to suppress the output from `mgen`, and we create variables with predictions at each level of `agecat`. The results are then plotted using a single `graph` command:

```
. quietly mgen, at(inc=(0(10)100) agecat=1) atmeans stub(A30) predlabel(Age 30-39)
. quietly mgen, at(inc=(0(10)100) agecat=2) atmeans stub(A40) predlabel(40-49)
. quietly mgen, at(inc=(0(10)100) agecat=3) atmeans stub(A50) predlabel(50+)
. graph twoway connected A30pr A40pr A50pr A50inc,
>   ytitle("Pr(In Labor Force)") xtitle("Income")
>   legend(cols(3))
```



Our example uses `graph` options to label the axes and improve the appearance of the legend. A brief discussion of `graph`'s options is included in chapter 2. For a more detailed discussion of the `graph` command, see the *Stata Graphics Reference Manual* or Mitchell (2012b).

Predictions over multiple outcome values

The greatest advantage of `mgen` over `marginsplot` occurs when you want to plot predictions for multiple outcomes (not multiple lines for the same outcome, but different outcomes) or to combine predictions from different models (for example, plot the predicted probabilities with different sets of control variables). Because a single `margins` command computes predictions for a single outcome category, predictions for multiple outcomes require running `margins` multiple times. `marginsplot` can only do plots based on running `margins` once. `mgen` does this automatically and produces variables named in a consistent fashion that makes plotting simple. As an example, we use a model from our chapter on ordinal outcomes:


```
. use gssclass4, clear
(gssclass4.dta | GSS Subjective Class Identification | 2013-11-20)
. ologit class i.fem i.white i.year i.ed age inc, nolog
(output omitted)
. mgen, at(age=(20(10)80)) stub(B) atmeans
Predictions from: margins, at(age=(20(10)80)) atmeans predict(outcome())
```

Variable	Obs	Unique	Mean	Min	Max	Label
Bpr1	7	7	.0481717	.0247463	.0799486	pr(y=lower) from margins
Bll1	7	7	.0425263	.0206772	.0705518	95% lower limit
Bul1	7	7	.053817	.0288155	.0893454	95% upper limit
Bage	7	7	50	20	80	age of respondent
BCpr1	7	7	.0481717	.0247463	.0799486	pr(y<=lower)
Bpr2	7	7	.4471604	.321645	.5647999	pr(y=working) from margins
Bll2	7	7	.4283944	.2954391	.544966	95% lower limit
Bul2	7	7	.4659264	.3478508	.5846338	95% upper limit
BCpr2	7	7	.4953321	.3463913	.6447485	pr(y<=working)
Bpr3	7	7	.484427	.3450458	.6195026	pr(y=middle) from margins
Bll3	7	7	.4645732	.3225423	.5927632	95% lower limit
Bul3	7	7	.5042807	.3675494	.646242	95% upper limit
BCpr3	7	7	.9797591	.9658939	.9897944	pr(y<=middle)
Bpr4	7	7	.020241	.0102057	.0341061	pr(y=upper) from margins
Bll4	7	7	.0166152	.0082046	.0276169	95% lower limit
Bul4	7	7	.0238667	.0122068	.0405953	95% upper limit
BCpr4	7	1	1	1	1	pr(y<=upper)

Specified values of covariates

1.	1.	2.	3.	2.	3.	
female	white	year	year	educ	educ	income
.5491103	.8140569	.4510676	.3099644	.5818505	.2414591	68.07737

To understand what `mgen` has done, we list some of the variables that were generated:

```
. list Bage Bpr1 Bpr2 Bpr3 Bpr4 in 1/8, clean
```

	Bage	Bpr1	Bpr2	Bpr3	Bpr4
1.	20	.0799486	.5647999	.3450458	.0102057
2.	30	.0660995	.5303931	.3910067	.0125007
3.	40	.0545074	.4917871	.4384017	.0153039
4.	50	.0448505	.4502864	.4861394	.0187237
5.	60	.0368379	.4072433	.533029	.0228899
6.	70	.0302114	.363968	.5778638	.0279568
7.	80	.0247463	.321645	.6195026	.0341061
8.

Variables `BPr1` through `BPr4` contain predicted probabilities for the four categories of class as they change by age. Most simply, we could graph these with `twoway line Bpr1 Bpr2 Bpr3 Bpr4 Bage`. Interpreting such graphs and making them more effective are topics discussed in subsequent chapters.

`mgen`'s defaults for handling multiple outcome values depend on model type. The way `mgen` distinguishes model types is by what options `predict` supports for a model. The following table summarizes the major types of models.

Model type	<code>predict</code> supports	Examples of models
Binary	<code>pr</code>	<code>logit</code> , <code>probit</code> (chapters 5 and 6)
Categorical	<code>outcome(#)</code>	<code>ologit</code> , <code>mlogit</code> (chapters 7 and 8)
Count	<code>pr(#)</code>	<code>poisson</code> , <code>nbreg</code> (chapter 9)
Other	None of the above	<code>regress</code>

For categorical models, *mgen* automatically computes predictions for all outcomes. The predicted probabilities for outcome *#* are named *stubpr#*. The cumulative probabilities $\Pr(y \leq \# | \mathbf{x})$ are also computed and named *stubCpr#*.

As with *mtable*, the *mgen* options `outcome()` and `pr()` can be used to select the outcomes for which predictions should be computed. For categorical models in which `predict` supports `outcome()`, `outcome(1 2)` will compute outcomes for categories 1 and 2. Cumulative probabilities will be calculated only if the set of specified values is the lowest observed values of *y*. For example, if the values of the outcome are 1, 2, and 3, `outcome(1 2)` will produce cumulative probabilities but `outcome(1 3)` will not. For binary models, `outcome(0 1)` computes predictions for both outcomes $y = 1$ and $y \neq 0$, whereas by default *mgen* only computes predictions for outcome $y = 1$. For count models, `pr()` can include a *numlist*. You can specify `pr(0/9)` to obtain predictions for values of *y* from 0 to 9. Cumulative probabilities are produced only if `pr()` specifies consecutive integers starting with the lowest observed value of *y*.

Observed and average predicted proportion using *mgen*, *meanpred*

By default, *mgen* generates variables where each row is a prediction from *margins* based on specified values of the independent variables. When option *meanpred* is used, *mgen* also generates variables in which the rows correspond to values of the outcome. These variables allow you to compare observed proportions versus average predicted probabilities, an important tool for count models in chapter 9. Variables generated by *mgen*, *meanpred* are as follows:

Variable name	Content
<i>stubval</i>	Value of category <i>k</i>
<i>stubobeq</i>	Observed proportion $y = k$
<i>stuboble</i>	Observed proportion $y \leq k$
<i>stubpreq</i>	Predicted proportion $y = k$
<i>stubprle</i>	Predicted proportion $y \leq k$
<i>stubobpr</i>	Difference between observed and predicted $y = k$
<i>stubcpreq</i>	Predicted probability $y = k$ given $y > 0$
<i>stubcprle</i>	Predicted probability $y = k$ given $y > 0$

Other mgen options

`atvars(varlist)` specifies the independent variables for which variables should be generated. `_none` indicates no variables. The default is to generate variables for all independent variables whose values vary over `at()`.

`noci` and `allstats` modify which variables are generated. When `noci` is used, no variables for confidence intervals are generated. When `allstats` is used, variables for the p -value (`stubpval`), z statistic (`stubz`), and standard error (`stubse`) are generated along with the prediction and confidence interval.

`level(#)` sets the level of the confidence interval from 10 to 99.99.

`predname(predname)` specifies the name of the variable (plus the `stub`) used for predictions generated by `margins`. By default, this is `pr` for probabilities and is `margins` otherwise.

`predlabel(string)` is used in the variable label for the prediction. This option can be useful for labeling variables that are being plotted.

`no label` indicates that value labels are not to be used in labeling generated variables.

`valuelength()` changes the length at which labels are truncated.

`conditional` is used to compute conditional rather than unconditional predictions for count models. This will be discussed in chapter 9.

4.7 Interpretation of parameters

Although the primary methods of interpretation in this book are based on predictions from the model, some methods of interpretation involve simple transformations of the model's parameters. For some estimation commands, there are options to list transformations of the estimates, such as the `or` option to list odds ratios for `logit` or the `beta` option to list standardized coefficients for `regress`. Although Stata is commendably clear in explaining the meaning of the estimated parameters, in some models it is easy to be confused about proper interpretations. For example, the `zip` model (discussed in chapter 9) simultaneously fits a binary and a count model, and it is easy to be confused regarding the direction of the effects.

For the estimation commands considered in this book, plus some not considered here, our `listcoef` command lists estimated coefficients in ways that facilitate interpretation. You can list coefficients selected by name or by significance level, list transformations of the coefficients, and request help on interpretation. In fact, often you will not need the normal output from the estimation. You could suppress this output with the prefix `quietly` (for example, `quietly logit lfp k5 wc hc`) and then use the `listcoef` command.

4.7.1 The listcoef command

The abbreviated syntax is

```
listcoef [varlist] [, [factor|percent|std] adjacent gt lt negative
         positive pvalue(#) nolabel constantoff help]
```

where *varlist* indicates that coefficients for only these variables are to be listed. If no *varlist* is given, then coefficients for all variables are listed. The *varlist* should not use factor-variable notation. For example, for the model `logit lfp i.agecat i.wc lwg`, the command `listcoef agecat` will show the coefficients for 2.agecat and 3.agecat. If `agecat##c.lwg` was in the model, estimates for all coefficients that include agecat would be listed.

Options for types of coefficients

Depending on the model and the specified options, `listcoef` computes standardized coefficients, factor changes in the odds or expected counts, or percentage changes in the odds or expected counts. More information on these types of coefficients is provided below, as well as in the chapters that deal with specific types of outcomes.

factor requests factor change coefficients indicating how many times larger or smaller the outcome is. In some cases, these coefficients are odds ratios.

percent requests percentage change coefficients indicating the percentage change in the outcome.

std requests that coefficients be standardized to a unit variance for the independent variables or the dependent variable. For models that can be derived from a latent-dependent variable (for example, the binary logit model), the variance of the latent outcome is estimated.

The following options (details on these options are given below) are available for each estimation command. If an option is the default, it does not need to be specified.

		std	factor	percent
Type 1:	mprobit, oprobit, probit, regress	Default	No	No
Type 2:	logistic, logit, ologit	Yes	Default	Yes
Type 3:	mlogit, nbreg, poisson, slogit, tnbreg, tpoisson, zinb, zip	No	Default	Yes

Options for `mlogit`, `mprobit`, and `slogit`

For the `mlogit`, `mprobit`, and `slogit` commands discussed in chapter 8, `listcoef` can show the coefficients for each pair of outcome categories. When these models are used with ordered outcomes, it is helpful to look at a subset of these coefficients. The following options are for this purpose:

adjacent specifies that only the coefficients from comparisons in which the two category values are adjacent will be printed (for example, comparing outcome 1 versus 2, and 2 versus 1, but not 1 versus 3). This option can be combined with **gt** or **lt**.

gt specifies that only the coefficients from comparisons in which the first category has a larger value than the second will be printed (for example, comparing outcome 2 versus 1, but not 1 versus 2).

lt specifies that only the coefficients from comparisons in which the first category has a smaller value than the second will be printed (for example, comparing outcome 1 versus 2, but not 2 versus 1).

negative specifies that only negative coefficients be shown. This option cannot be combined with **adjacent**, **gt**, or **lt**.

positive specifies that only positive coefficients be shown. This option cannot be combined with **adjacent**, **gt**, or **lt**.

Other options

pvalue(#) specifies that only coefficients significant at the # significance level or smaller will be printed. For example, **pvalue(.05)** specifies that only coefficients significant at the 0.05 level should be listed. If **pvalue()** is not given, all coefficients are listed.

nolabel requests that category numbers rather than value labels be used in the output.

constantoff specifies to not include the constant(s) in the output. By default, they are listed. In Stata 10 and earlier, the default is **constantoff**, and you must use the option **constant** to list the constants.

help gives details for interpreting each coefficient.

4.7.2 Standardized coefficients

std requests coefficients after variables have been standardized to a unit variance. Standardized coefficients are computed as follows.

x-standardized coefficients. The linear regression model can be expressed as

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon \quad (4.2)$$

Let σ_k be the standard deviation of x_k . Then dividing each x_k by σ_k and multiplying the corresponding β_k by σ_k becomes

$$y = \beta_0 + (\sigma_1\beta_1) \frac{x_1}{\sigma_1} + (\sigma_2\beta_2) \frac{x_2}{\sigma_2} + \varepsilon$$

$\beta_k^{S_x} = \sigma_k\beta_k$ is an x -standardized coefficient. For a continuous variable, $\beta_k^{S_x}$ can be interpreted as follows:

For a standard deviation increase in x_k , y is expected to change by $\beta_k^{S_x}$ units, holding other variables constant.

The same method of x -standardization can be used in all the other models we consider in this book.

y - and y^* -standardized coefficients. To standardize the dependent variable, let σ_y be the standard deviation of y . We standardize y by dividing (4.2) by σ_y :

$$\frac{y}{\sigma_y} = \frac{\beta_0}{\sigma_y} + \frac{\beta_1}{\sigma_y} x_1 + \frac{\beta_2}{\sigma_y} x_2 + \frac{\varepsilon}{\sigma_y}$$

Then $\beta_k^{S_y} = \beta_k/\sigma_y$ is a y -standardized coefficient that can be interpreted as follows:

For a unit increase in x_k , y is expected to change by $\beta_k^{S_y}$ standard deviations, holding other variables constant.

For a binary independent variable,

Having characteristic x_k as opposed to not having the characteristic results in an expected change in y of $\beta_k^{S_y}$ standard deviations, holding other variables constant.

Or more simply,

Having characteristic x_k results in an expected change in y of $\beta_k^{S_y}$ standard deviations, holding other variables constant.

In models with a latent dependent variable, the equation $y^* = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$ can be divided by $\hat{\sigma}_{y^*}$. To estimate the variance of the latent variable, the quadratic form is used:

$$\widehat{\text{Var}}(y^*) = \hat{\beta}' \widehat{\text{Var}}(\mathbf{x}) \hat{\beta} + \text{Var}(\varepsilon)$$

where $\hat{\beta}$ is a vector of estimated coefficients and $\widehat{\text{Var}}(\mathbf{x})$ is the covariance matrix for the x 's computed from the observed data. By assumption, $\text{Var}(\varepsilon) = 1$ in probit models and $\text{Var}(\varepsilon) = \pi^2/3$ in logit models.

Fully standardized coefficients. In the linear regression model, it is possible to standardize both y and the x 's:

$$\frac{y}{\sigma_y} = \frac{\beta_0}{\sigma_y} + \left(\frac{\sigma_1 \beta_1}{\sigma_y} \right) \frac{x_1}{\sigma_1} + \left(\frac{\sigma_2 \beta_2}{\sigma_y} \right) \frac{x_2}{\sigma_2} + \frac{\varepsilon}{\sigma_y}$$

Then $\beta_k^S = (\sigma_k \beta_k) / \sigma_y$ is a fully standardized coefficient that can be interpreted as follows:

For a standard deviation increase in x_k , y is expected to change by β_k^S standard deviations, holding other variables constant.

The same approach can be used in models with a latent-dependent variable y^* .

Example of listcoef for standardized coefficients

Here we illustrate the computation of standardized coefficients for the regression model. Examples for other models are given in later chapters. The standard output from regress is

```
. use science4, clear
(science4.dta | Long's scientific career data | 2014-03-02)
. regress job i.female i.phdclass mentcit3yr fellow publ citi
```

Source	SS	df	MS				
Model	30.5973982	8	3.82467477	Number of obs = 161			
Residual	94.0515545	152	.618760227	F(8, 152) = 6.18			
Total	124.648953	160	.779055954	Prob > F = 0.0000			
				R-squared = 0.2455			
				Adj R-squared = 0.2058			
				Root MSE = .78661			

job	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
female						
yes	-.0886385	.1655774	-0.54	0.593	-.4157688	.2384918
phdclass						
good	.4003174	.220298	1.82	0.071	-.0349241	.8355589
strong	.8089664	.2297963	3.52	0.001	.3549592	1.262974
elite	.8871308	.236539	3.75	0.000	.4198021	1.35446
mentcit3yr	.0023816	.0023724	1.00	0.317	-.0023056	.0070688
fellow	.1947417	.1328996	1.47	0.145	-.0678272	.4573106
publ	.0004072	.0256824	0.02	0.987	-.0503334	.0511477
citi	.0076907	.0041223	1.87	0.064	-.0004538	.0158351
_cons	2.062959	.2140085	9.64	0.000	1.640144	2.485775

If we use `listcoef`, we get

```
. listcoef, help
regress (N=161): Unstandardized and standardized estimates
Observed SD: 0.8826
SD of error: 0.7866
```

	b	t	P> t	bStdX	bStdY	bStdXY	SDofX
female							
yes	-0.0886	-0.535	0.593	-0.038	-0.100	-0.043	0.430
phdclass							
good	0.4003	1.817	0.071	0.181	0.454	0.206	0.453
strong	0.8090	3.520	0.001	0.362	0.917	0.410	0.447
elite	0.8871	3.750	0.000	0.418	1.005	0.474	0.471
mentcit3yr	0.0024	1.004	0.317	0.075	0.003	0.085	31.644
fellow	0.1947	1.465	0.145	0.098	0.221	0.111	0.501
pub1	0.0004	0.016	0.987	0.001	0.000	0.001	3.228
cit1	0.0077	1.866	0.064	0.163	0.009	0.185	21.242
constant	2.0630	9.640	0.000

```
b = raw coefficient
t = t-score for test of b=0
P>|t| = p-value for t-test
bStdX = x-standardized coefficient
bStdY = y-standardized coefficient
bStdXY = fully standardized coefficient
SDofX = standard deviation of X
```

By default for `regress`, `listcoef` lists standardized coefficients for all variables. If we are interested in listing coefficients for a subset of variables, we can specify a *varlist* after `listcoef`. For factor variables, you should specify only the source name, not the factor-variable notation (for example, `i.female`) or the name of the variable that is constructed (for example, `2.phdclass`). Here is an example:

```
. listcoef female phdclass pub1
regress (N=161): Unstandardized and standardized estimates
Observed SD: 0.8826
SD of error: 0.7866
```

	b	t	P> t	bStdX	bStdY	bStdXY	SDofX
female							
yes	-0.0886	-0.535	0.593	-0.038	-0.100	-0.043	0.430
phdclass							
good	0.4003	1.817	0.071	0.181	0.454	0.206	0.453
strong	0.8090	3.520	0.001	0.362	0.917	0.410	0.447
elite	0.8871	3.750	0.000	0.418	1.005	0.474	0.471
pub1	0.0004	0.016	0.987	0.001	0.000	0.001	3.228

4.7.3 Factor and percentage change coefficients

In logit models and count models, coefficients can be expressed in two ways:

1. Coefficients can indicate the factor or multiplicative change in the odds, relative risks, or expected count. These are the default for some models or can be requested with the **factor** option with **listcoef**.
2. Percent changes in these quantities can be requested with the **percent** option.

Details on these coefficients are given in later chapters for each specific model.

4.8 Next steps

This concludes our discussion of the basic commands and options that are used for fitting, testing, assessing fit, and interpreting regression models. In the next five chapters, we show how these commands can be applied for models for different types of outcomes. Although chapters 5 and 6 have more detail than later chapters, you should be able to proceed from here to any of the chapters that follow.

Part II

Models for specific kinds of outcomes

In part II, we provide information on the models appropriate for different kinds of dependent outcomes.

- **Chapters 5 and 6** consider models for binary outcomes. These models provide the foundation for the models for other types of outcomes in the rest of the book. For this reason, we provide more detailed explanations than in later chapters. We divide the material into two chapters. Chapter 5 shows how to fit the binary regression model, how to test hypotheses, how to compute residuals and influence statistics, and how to calculate scalar measures of model fit. Chapter 6 focuses entirely on how these models can be interpreted using predicted probabilities, marginal effects, and odds ratios. We recommend that all readers review these chapters, even if you are interested mainly in other types of outcomes.

In contrast, chapters 7, 8, and 9 can be read in any combination or order, depending on your interests. Each chapter provides information on fitting the relevant models, testing hypotheses about the coefficients, and interpretation in terms of predicted probabilities. In addition,

- **Chapter 7** on ordinal outcomes describes the parallel regression assumption that is made by the ordered logit and probit models and shows how this assumption can be tested. We also discuss interpretation in terms of the underlying latent variable and odds ratios. Most of the methods and commands for interpretation for ordinal models can be applied to models for nominal outcomes, so those readers primarily interested in nominal outcomes should at least review this chapter.
- **Chapter 8** on nominal outcomes introduces the multinomial logit model. We discuss the assumption of the independence of irrelevant alternatives and present two graphical methods of interpretation. Methods of interpretation from chapter 7 are extended, and some new methods are introduced. We also briefly discuss and then consider the multinomial probit model without correlated errors and the stereotype logistic regression model. We briefly consider models for nominal outcomes with alternative-specific data, such as the conditional logit and multinomial probit models.

- **Chapter 9** on count outcomes begins with the Poisson and negative binomial regression models. We show how to test the Poisson model's assumption of equidispersion and how to incorporate differences in exposure time into the models. The next two models, the zero-truncated Poisson and negative binomial models, deal with the common problem of having no zeros in your data. We combine these models with the logit model to construct the hurdle model for counts. We conclude by considering two zero-inflated models that are designed for data with an "excess" of zero counts.

5 Models for binary outcomes: Estimation, testing, and fit

Binary outcomes are ubiquitous and examples come easily to mind. Did a person vote? Is a manufacturing firm unionized? Does someone consider themselves a feminist or not? Did a startup company go bankrupt? Does a person have arthritis? This chapter focuses on the two most often used models for binary outcomes, the binary logit and binary probit models, referred to jointly as the binary regression model (BRM). The BRM allows a researcher to explore how each explanatory variable affects the probability of the event occurring.

The BRM is also the foundation from which more complex models for ordinal, nominal, and count models are derived. Ordinal and nominal regression models are equivalent to simultaneously fitting a set of BRMs. Although the link is less direct in count models, the Poisson distribution can be derived as the outcome of many binary trials. Consequently, the principles of fitting, testing, and interpreting binary models provide essential tools that are used in later chapters. Although each chapter of the book is largely self-contained, the two chapters on binary outcomes provide more detailed explanations than later chapters. As a result, even if your interests are in models for ordinal, nominal, or count outcomes, you will benefit from reading this chapter and the next one.

We begin the chapter by reviewing the mathematical structure of the binary regression model. We then examine statistical testing and fit. These discussions are brief, and much of it is intended either as a simple overview or as a review for those who are familiar with the models. For a complete discussion, see Agresti (2013), Hosmer, Lemeshow, and Sturdivant (2013), or Long (1997). Although the material in this chapter is fundamental to working with these models, we anticipate that the more important contribution of this book will be in helping you interpret and present results. The issues involved in effective interpretation are extensive enough that we devote a chapter of its own to the topic, to which this chapter might be considered the prelude.

5.1 The statistical model

There are three ways to derive the BRM, with each method leading to the same statistical model. First, a latent variable can be hypothesized along with a measurement model relating the latent variable to the observed binary outcome. Second, the model can be constructed as a probability model. Third, the model can be generated as a random

utility or discrete choice model. This last approach is not considered in our review; see Long (1997, 155–156) for an introduction and Train (2009) for a detailed discussion.

5.1.1 A latent-variable model

Assume a latent or unobserved variable y^* ranging from $-\infty$ to ∞ that is related to the observed independent variables by the structural model

$$y_i^* = \mathbf{x}_i\beta + \varepsilon_i$$

where i indicates the observation and ε is a random error. For a single independent variable, we can simplify the notation to

$$y_i^* = \alpha + \beta x_i + \varepsilon_i$$

These equations are identical to those for the linear regression model except—and this is a big exception—that the dependent variable is unobserved.

The observed binary dependent variable has two values, typically coded as 0 for a negative outcome (that is, the event did not occur) and 1 for a positive outcome (that is, the event did occur). A measurement equation defines the link between the binary observed variable y and the continuous latent variable y^* :

$$y_i = \begin{cases} 1 & \text{if } y_i^* > 0 \\ 0 & \text{if } y_i^* \leq 0 \end{cases}$$

Cases with positive values of y^* are observed as $y = 1$, while cases with negative or 0 values of y^* are observed as $y = 0$.

To give a concrete example, imagine a survey item that asks respondents if they agree or disagree with the proposition that “a working mother can establish just as warm and secure a relationship with her children as a mother who does not work”. Obviously, respondents will vary greatly in their opinions. Some people adamantly agree with the proposition, some adamantly disagree, and still others have weak opinions one way or the other. Imagine an underlying continuum y^* of feelings about this item, with each respondent having a specific value on the continuum. Those respondents with positive values for y^* will answer “agree” to the survey question ($y = 1$) and those with negative values will “disagree” ($y = 0$). A shift in a respondent’s opinion might move her from agreeing strongly with the position to agreeing weakly with the position, which would not change the response we observe. Or, the respondent might move from weakly agreeing to weakly disagreeing, in which case, we would observe a change from $y = 1$ to $y = 0$.

Consider a second example, which we use throughout this chapter. Let $y = 1$ if a woman is in the paid labor force and let $y = 0$ if she is not. The independent variables include age, number of children, education, family income, and expected wages. Not all women in the labor force ($y = 1$) are there with the same certainty. One woman might be close to leaving the labor force, whereas another woman could be firm in her decision

to work. In both cases, we observe $y = 1$. The idea of a latent y^* is that an underlying propensity to work generates the observed state. Although we cannot directly observe the propensity, at some point a change in y^* results in a change in what we observe, namely, whether the woman is in the labor force.

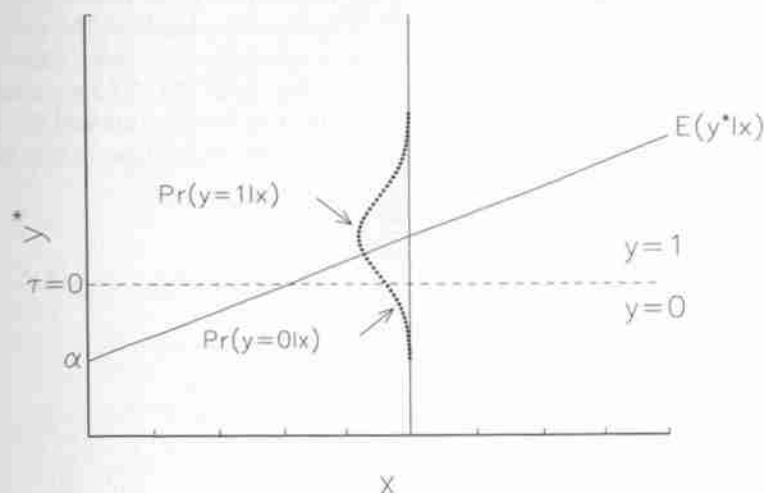


Figure 5.1. Relationship between latent variable y^* and $\Pr(y = 1)$ for the BRM

The latent-variable model for a binary outcome with a single independent variable is shown in figure 5.1. For a given value of x ,

$$\Pr(y = 1 | x) = \Pr(y^* > 0 | x)$$

Substituting the structural model and rearranging terms,

$$\Pr(y = 1 | x) = \Pr(\varepsilon > -[\alpha + \beta x] | x) \quad (5.1)$$

which shows how the probability depends on the distribution of the error ε .

Two distributions of ε are commonly used, both with an assumed mean of 0. First, ε is assumed to be normal with $\text{Var}(\varepsilon) = 1$. This leads to the binary probit model in which (5.1) becomes

$$\Pr(y = 1 | x) = \int_{-\infty}^{\alpha + \beta x} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) dt$$

Alternatively, ε is assumed to be distributed logistically with $\text{Var}(\varepsilon) = \pi^2/3$, leading to the binary logit model with the simpler equation

$$\Pr(y = 1 | x) = \frac{\exp(\alpha + \beta x)}{1 + \exp(\alpha + \beta x)} \quad (5.2)$$

The peculiar value assumed for $\text{Var}(\varepsilon)$ in the logit model illustrates a basic point about the identification of models with latent outcomes. In the linear regression model, $\text{Var}(\varepsilon)$ can be estimated because y is observed. In the BRM, $\text{Var}(\varepsilon)$ cannot be estimated because the dependent variable is unobserved. Accordingly, the model is unidentified unless an assumption is made about the variance of the errors. For probit, we assume $\text{Var}(\varepsilon) = 1$ because this leads to a simple form of the model. If a different value was assumed, this would simply change the values of the structural coefficients uniformly. In the logit model, the variance is set to $\pi^2/3$ because this leads to the simple form in (5.2). Although the value assumed for $\text{Var}(\varepsilon)$ is arbitrary, the value chosen does not affect the computed value of the probability (see Long [1997, 49–50] for a simple proof). Changing the assumed variance affects the spread of the distribution and the magnitude of the regression coefficients, but it does not affect the proportion of the distribution above or below the threshold.

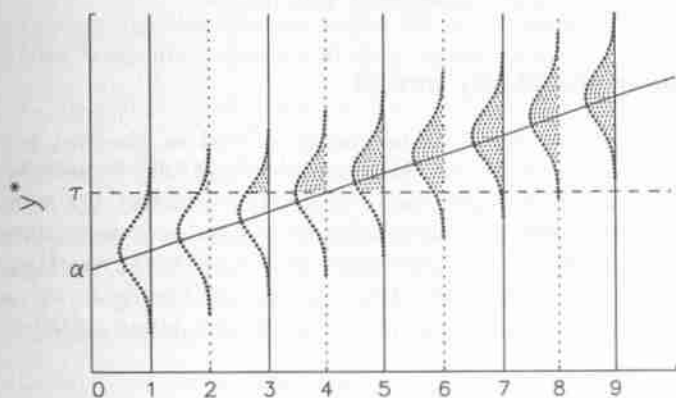
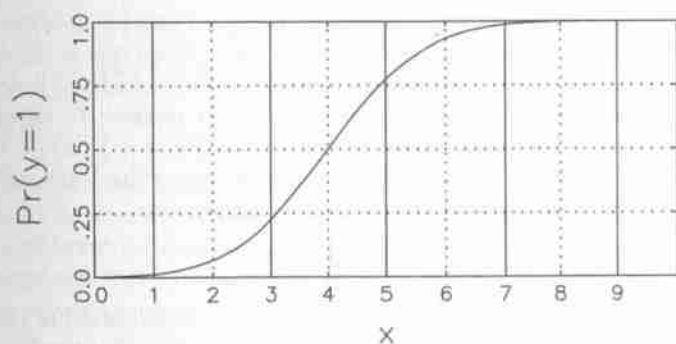
Panel A: Plot of y^* Panel B: Plot of $\Pr(y=1|x)$ 

Figure 5.2. Relationship between the linear model $y^* = \alpha + \beta x + \varepsilon$ and the nonlinear probability model $\Pr(y = 1 | x) = F(\alpha + \beta x)$

For both logit and probit, the probability of the event conditional on \mathbf{x} is the cumulative density function (CDF) of ε evaluated at $\mathbf{x}\beta$,

$$\Pr(y = 1 | \mathbf{x}) = F(\mathbf{x}\beta)$$

where F is the normal CDF Φ for the probit model and the logistic CDF Λ for the logit model. The relationship between the linear latent-variable model and the resulting nonlinear probability model is shown in figure 5.2 for a model with one independent variable. Panel A shows the error distribution for nine values of x . The area where $y^* > 0$ corresponds to $\Pr(y = 1 | x)$ and has been shaded. Panel B plots $\Pr(y = 1 | x)$ corresponding to the shaded regions in panel A. As we move from 1 to 2, only a portion of the thin tail crosses the threshold in panel A, resulting in a small change in $\Pr(y = 1 | x)$ in panel B. As we move from 2 to 3 to 4, thicker regions of the error distribution slide

over the threshold, and the increase in $\Pr(y = 1 | x)$ becomes larger. The resulting curve is the well-known S-curve associated with the BRM.

5.1.2 A nonlinear probability model

Can all binary dependent variables be conceptualized as observed manifestations of some underlying latent propensity? Although philosophically interesting, perhaps, the question is of little practical importance, because the BRM can also be derived without appealing to a latent variable. This is done by specifying a nonlinear model relating the x 's to the probability of an event. Following Theil (1970), the logit model can be derived by constructing a model in which the predicted $\Pr(y = 1 | \mathbf{x})$ is forced to be within the range 0 to 1. For example, in the linear probability model,

$$\Pr(y = 1 | \mathbf{x}) = \mathbf{x}\beta + \varepsilon$$

the predicted probabilities can be greater than 1 and less than 0. To constrain the predictions to the range 0 to 1, we first transform the probability into the odds,

$$\Omega(\mathbf{x}) = \frac{\Pr(y = 1 | \mathbf{x})}{\Pr(y = 0 | \mathbf{x})} = \frac{\Pr(y = 1 | \mathbf{x})}{1 - \Pr(y = 1 | \mathbf{x})}$$

which indicate how often something happens ($y = 1$) relative to how often it does not happen ($y = 0$). The odds range from 0 when $\Pr(y = 1 | \mathbf{x}) = 0$ to ∞ when $\Pr(y = 1 | \mathbf{x}) = 1$. The log of the odds, often referred to as the logit, ranges from $-\infty$ to ∞ . This range suggests a model that is linear in the logit:

$$\ln \Omega(\mathbf{x}) = \mathbf{x}\beta$$

This equation is equivalent to the logit model (5.2). Interpretation of this form of the model often focuses on factor changes in the odds, which are discussed below.

Other binary regression models are created by choosing functions of $\mathbf{x}\beta$ that range from 0 to 1. CDFs have this property and readily provide several examples. For example, the CDF for the standard normal distribution results in the probit model.

5.2 Estimation using logit and probit commands

Logit and probit models can be fit with the following commands and their basic options:

```
logit depvar [indepvars] [if] [in] [weight] [, noconstant asis or
      vce(vcetype) ]
```

```
probit depvar [indepvars] [if] [in] [weight] [, noconstant asis
      vce(vcetype) ]
```


Variable lists

depvar is the dependent variable. *indepvars* is a list of independent variables. If *indepvars* is not included, Stata fits a model with only an intercept.

Warning about dependent variable. In binary models, all nonmissing, nonzero values of *depvar* are classified as positive outcomes, traditionally referred to as successes. Only zero values are considered negative outcomes, which are referred to as failures. Because negative values are nonzero, they are considered to be positive outcomes. To avoid possible confusion, we recommend that you explicitly create a 0/1 variable for use as *depvar*.

Specifying the estimation sample

if and in qualifiers. *if* and *in* qualifiers can be used to restrict the estimation sample. For example, if you want to fit a logit model for only women who went to college, as indicated by the variable *wc*, you could specify `logit lfp k5 k618 age hc lwg if wc==1`.

Listwise deletion. Stata excludes cases in which there are missing values for any of the variables in the model. Accordingly, if two models are fit using the same dataset but have different independent variables, the models may have different samples. We recommend that you use *mark* and *markout* (discussed in section 3.1.6) to explicitly remove cases with missing data.

Weights and complex samples

Both *logit* and *probit* can be used with *fweight*, *pweight*, and *iweight*. Survey estimation can be done using *svy: logit* or *svy: probit*. See section 3.1.7 for details.

Options

noconstant specifies that the model not have a constant term.

asis specifies that estimates for variables that have perfect prediction should be included in the results table. For details, see page 197.

or (for *logit* only) reports the odds ratios defined as $\exp(\hat{\beta})$. Standard errors and confidence intervals are similarly transformed. Alternatively, our *listcoef* command can be used.

vce(vcetype) specifies the type of standard errors to be computed. See section 3.1.9 for details.

5.2.1 Example of logit model

Our example is from Mroz's (1987) study of the labor force participation of women, using data from the 1976 Panel Study of Income Dynamics.¹ The sample consists of 753 white, married women between the ages of 30 and 60 years. The dependent variable `lfp` equals 1 if a woman is in the labor force and equals 0 otherwise. We use `codebook`, `compact` to list information about the variables we plan to include in our model:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. codebook lfp k5 k618 agecat wc hc lwg inc, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
<code>lfp</code>	753	2	.5683931	0	1	In paid labor force?
<code>k5</code>	753	4	.2377158	0	3	# kids < 6
<code>k618</code>	753	9	1.353254	0	8	# kids 6-18
<code>agecat</code>	753	3	1.823373	1	3	Wife's age group
<code>wc</code>	753	2	.2815405	0	1	Wife attended college?
<code>hc</code>	753	2	.3917663	0	1	Husband attended college?
<code>lwg</code>	753	676	1.097115	-2.054124	3.218876	Log of wife's estimated...
<code>inc</code>	753	621	20.12897	-.0290001	96	Family income excluding...

Although the meaning of most of the variables is clear from the label, `lwg` is the log of an estimate of what the wife's wages would be if she was employed, given her other characteristics. Because the outcome is labor force participation, it is important to include what the wife might be expected to earn if she was employed. Following the same reasoning, `inc` is family income excluding whatever the wife earns; this is, therefore, a measure of what the family income would be if the wife was not employed. We consider interpretation later, but it may also help bearing in mind that the data are from 1976. In the United States, prices have risen by just over a factor of 4 between 1976 and 2014, so a change in income of \$5,000 in 1974 is similar to a change in income of \$20,000 in 2014.

Because `agecat` is ordinal, we use `tabulate` to examine the distribution among the age groups:

```
. tabulate agecat, missing
```

Wife's age group	Freq.	Percent	Cum.
30-39	298	39.58	39.58
40-49	290	38.51	78.09
50+	165	21.91	100.00
Total	753	100.00	

1. These data were generously made available by Thomas Mroz.

Next, we want to fit the logit model. To be consistent with the naming practice Stata will use, we use `2.agecat` and `3.agecat` to refer to dummy variables indicating whether `agecat==2` and whether `agecat==3`, respectively. By fitting the logit model,

$$\Pr(\text{lfp} = 1) = F(\beta_0 + \beta_{k5}k5 + \beta_{k618}k618 + \beta_{2.agecat}2.agecat + \beta_{3.agecat}3.agecat + \beta_{wc}wc + \beta_{hc}hc + \beta_{lwglwg} + \beta_{inc}inc)$$

we obtain the following results:

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc
```

```
Iteration 0:  log likelihood = -514.8732
Iteration 1:  log likelihood = -453.10297
Iteration 2:  log likelihood = -452.72408
Iteration 3:  log likelihood = -452.72367
Iteration 4:  log likelihood = -452.72367
```

Logistic regression	Number of obs	=	753
	LR chi2(8)	=	124.30
	Prob > chi2	=	0.0000
	Pseudo R2	=	0.1207

Log likelihood = -452.72367

	lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
	k5	-1.391567	.1919279	-7.25	0.000	-1.767739 -1.015395
	k618	-.0656678	.068314	-0.96	0.336	-.1995607 .0682251
	agecat					
	40-49	-.6267601	.208723	-3.00	0.003	-1.03585 -0.2176705
	50+	-1.279078	.2597827	-4.92	0.000	-1.788242 -.7699128
	wc					
	college	.7977136	.2291814	3.48	0.001	.3485263 1.246901
	hc					
	college	.1358895	.2054464	0.66	0.508	-.266778 .5385569
	lwg	.6099096	.1507975	4.04	0.000	.314352 .9054672
	inc	-.0350542	.0082718	-4.24	0.000	-.0512666 -.0188418
	_cons	1.013999	.2860488	3.54	0.000	.4533539 1.574645

The information in the header and table of coefficients is in the same form as discussed in chapter 3. The iteration log begins with `Iteration 0: log likelihood = -514.8732` and ends with `Iteration 4: log likelihood = -452.72367`, with the intermediate iterations showing the steps taken in the numerical maximization of the log-likelihood function. Although this information can provide insights when the model does not converge, in our experience it is of little use in logit and probit, where we have never seen problems with convergence. Accordingly, when fitting further models, we use the `nolog` option to suppress the log. If a model does not converge or the estimates seem “off”, we would rerun the model without the `nolog` option.

We use factor-variable notation for the categorical variable `agecat`, as well as for the binary variables `wc` and `hc`. We discussed factor-variable notation in detail in chapter 3. Using factor-variable notation for binary variables in this case may seem unnecessary, because we get the same coefficients regardless; but for some of the interpretation

techniques we demonstrate later, specifying `i.wc` instead of just `wc` in the estimation command is essential to obtain proper results (see section 6.2). As a result, we find it good practice to always enter binary variables into our models with the `i.` syntax.

In the output above, estimates for these variables are labeled on two lines with the first line indicating the name of the variable (for example, `wc`) and the second line listing the value label for category 1 (in this case, `college`). If you are using Stata 12 or earlier, the category value is printed instead of the value label. This requires you to ensure that you know the meanings of the category values. For users with Stata 13 and later, we recommend always using value labels with factor variables to avoid confusion.

Also, by default, the reference or base category of a factor variable is not listed. Adding the `allbase` option to an estimation command will display the base reference category. For `agecat` in the above example, the value 1 is the base category because this is the first value; specifying `ib3.agecat` would have used `agecat==3` as the base category instead.

5.2.2 Comparing logit and probit

Above, we fit the model with `logit`, but we could have used `probit` instead. An easy way to show how the results would differ is to put them side by side in a single table. We can do this by using `estimates table` (see [R] [estimates table](#)), which is more generally useful for combining results from multiple models into one table. After fitting the logit model, we use `estimates store estname` to save the estimates with the name `Mlogit`:

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
      (output omitted)
. estimates store Mlogit
```

We then fit a probit model and store the results:

```
. probit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
      (output omitted)
. estimates store Mprobit
```

Next, we combine the results with `estimates table`. Option `b()` sets the format for displaying the coefficients. `b(%9.3)` lists the estimates in nine columns with five decimal places. Option `t` requests test statistics for individual coefficients—either z tests or t tests depending on the model that was fit.² `varlabel` uses variable labels rather than variable names to label coefficients (the option was named `label` before Stata 13), with `varwidth()` indicating how many columns should be used for the labels. Variable names and value labels are used with factor variables.

2. The `estimates table` output labels the test statistic as `t` regardless of whether z tests or t tests are used.


```
. estimates table Mlogit Mprobit, b(%9.3f) t varlabel varwidth(30)
```

Variable	Mlogit	Mprobit
# kids < 6	-1.392	-0.840
	-7.25	-7.50
# kids 6-18	-0.066	-0.041
	-0.96	-1.01
agecat		
40-49	-0.627	-0.382
	-3.00	-3.06
50+	-1.279	-0.780
	-4.92	-5.00
wc		
college	0.798	0.482
	3.48	3.55
hc		
college	0.136	0.074
	0.66	0.60
Log of wife's estimated wages	0.610	0.371
	4.04	4.21
Family income excluding wife's	-0.035	-0.021
	-4.24	-4.37
Constant	1.014	0.622
	3.54	3.69

legend: b/t

Comparing results, the estimated logit coefficients are about 1.7 times larger than the probit estimates. For example, the ratios for `k5` and `inc` are 1.66. This illustrates how the magnitudes of the coefficients are affected by the assumed $\text{Var}(\varepsilon)$. The ratio of estimates for `hc` is larger because of the large standard errors for these estimates. Values of the z tests for logit and probit are quite similar because they are not affected by the assumed $\text{Var}(\varepsilon)$, but they are not exactly the same because the models assume different distributions of the errors.

5.2.3 (Advanced) Observations predicted perfectly

We mark this section as advanced because if you work with large samples where your outcome variable is not rare, you may never encounter perfect prediction. If you have smaller samples with binary predictors, you may encounter it regularly. We suggest you read enough of this section to understand what perfect prediction is so that you will recognize it if it occurs in your analysis.

Maximum likelihood estimation is not possible when the dependent variable does not vary within one of the categories of an independent variable. This is referred to as perfect prediction or quasicomplete separation. To illustrate this, suppose that we are treating

k5 as categorical rather than continuous in our model of labor force participation. To do this, we regress lfp on indicator variables for the number of children.³ Variable k5_1 equals 1 if a person had one young child and equals 0 otherwise, and so on for k5_2 and k5_3. Only three respondents had three young children, and none of these women were in the paid labor force:

```
. tabulate lfp k5
```

In paid labor force?	# kids < 6				Total
	0	1	2	3	
not in LF	231	72	19	3	325
in LF	375	46	7	0	428
Total	606	118	26	3	753

We find that lfp is 0 every time k5_3 is 0. A logit model predicting lfp with the binary variables k5_1, k5_2, and k5_3 (with no children being the excluded category) cannot be estimated because the observed coefficient for k5_3 is effectively infinite. Think of it this way: The observed odds of being in the labor force for those with no children is $375/231 = 1.62$, while the observed odds for those with three young children is $0/3 = 0$. The odds ratio is $0/1.62 = 0$. For the odds ratios to be 0, $\hat{\beta}_{k5_3}$ must be negative infinity. As the likelihood is maximized, estimates of β_{k5_3} get more and more negative until Stata realizes that the parameter cannot be estimated and reports the following:

```
. logit lfp k5_1 k5_2 k5_3, or nolog
note: k5_3 != 0 predicts failure perfectly
      k5_3 dropped and 3 obs not used
```

```
Logistic regression               Number of obs   =       750
                                LR chi2(2)         =       31.05
                                Prob > chi2         =       0.0000
                                Pseudo R2          =       0.0303

Log likelihood = -496.82164
```

lfp	Odds Ratio	Std. Err.	z	P> z	[95% Conf. Interval]	
k5_1	.3935556	.0812515	-4.52	0.000	.2625858	.5898491
k5_2	.2269474	.1021224	-3.30	0.001	.0939505	.5482153
k5_3	1 (omitted)					
_cons	1.623377	.1357794	5.79	0.000	1.377922	1.912555

The message

```
note: k5_3 != 0 predicts failure perfectly
      k5_3 dropped and 3 obs not used
```

can be interpreted as follows. If someone in the sample has three young children (that is, if $k5_3 \neq 0$), then she is never in the labor force (that is, $lfp = 0$), which is a “failure”

3. We could do this with the factor variable i.k5, but because we want to illustrate the use of `exlogistic`, which does not allow factor variables, we created our own indicator variables.

in the terminology of the model. At this point, Stata drops the three cases where `k5_3` is 0 and also drops `k5_3` from the model. In the output, the coefficient for `k5_3` is shown as 1 followed by (omitted). If you use the `asis` option, Stata keeps `k5_3` and the observations in the model and shows the estimate at convergence:

```
. logit lfp k5_1 k5_2 k5_3, or asis nolog
Logistic regression               Number of obs   =       753
                                LR chi2(2)          =       36.10
                                Prob > chi2         =       0.0000
                                Pseudo R2           =       0.0351

Log likelihood = -496.82164
```

	lfp	Odds Ratio	Std. Err.	z	P> z	[95% Conf. Interval]	
	k5_1	.3935556	.0812515	-4.52	0.000	.2625858	.5898491
	k5_2	.2269091	.102109	-3.30	0.001	.0939316	.548141
	k5_3	4.43e-10
	_cons	1.623377	.1357794	5.79	0.000	1.377922	1.912555

Note: 3 failures and 0 successes completely determined.

The estimated odds ratio of 4.43e-10 (that is, 0.000000000443) is Stata's attempt to estimate an odds ratio that is 0. With perfect prediction, the estimates for the other variables can be used, but you do not learn anything useful about the variable that is dropped.

Just because, in our sample, the three women with children under age 5 were not in the labor force does not imply the probability is 0 in the population. With any nonzero probability in the population, there is some chance that every observation with a given value of an independent variable will have the same outcome. This is especially so when the sample is small.

Exact methods of estimation provide more accurate inference in small samples than standard maximum likelihood estimation. In Stata, `exlogistic` provides exact method estimation for the logit model. Mehta and Patel (1995) provide an accessible review of these methods. Exact estimation computes *p*-values by enumerating all possible outcomes, which can provide estimates and significance levels in small samples with perfect prediction. However, computing all enumerations takes a long time. For example, fitting the logit model above with `exlogistic` took 750 times longer than fitting the model with `logit`. The results are


```
. exlogistic lfp k5_1 k5_2 k5_3, memory(2gb)
Enumerating sample-space combinations:
observation 1: enumerations =      2
(output omitted)
observation 430: enumerations = 2101996
(output omitted)
observation 753: enumerations = 12852
note: CMLE estimate for k5_3 is -inf; computing MUE
Exact logistic regression
```

Number of obs =	753
Model score =	35.00541
Pr >= score =	0.0000

lfp	Odds Ratio	Suff.	2*Pr(Suff.)	[95% Conf. Interval]	
k5_1	.3940829	46	0.0000	.2564596	.600809
k5_2	.2274839	7	0.0009	.0795388	.5759369
k5_3	.1610353*	0	0.1126	0	1.504537

(*) median unbiased estimates (MUE)

By default, `exlogistic` provides the conditional maximum likelihood estimates of parameters. When those estimates are infinite, as is the case with perfect prediction, median unbiased estimates are given. Cytel Software Corporation (2005, 512) suggests that the median unbiased estimates should be interpreted with caution by using the confidence interval. Although one bound of the confidence interval for the odds ratio will be 0 or positive infinity, the other bound is informative, and we can be 95% confident that the estimate is larger (or smaller) than this bound. This allows us to speak precisely about how much uncertainty we have. For example, in the output above, the confidence interval includes 1, so our data do not even permit us to reject the null hypothesis that women with three children are less likely to be in the labor force than are women with no children.

5.3 Hypothesis testing

Hypothesis tests of regression coefficients can be conducted with the z statistics from the estimation output, with the `test` command for Wald tests of simple and complex hypotheses, and with the `lrtest` command for the corresponding likelihood-ratio (LR) tests. We discuss using each to test hypotheses involving a single coefficient and then show how `test` and `lrtest` can be used for hypotheses involving multiple coefficients. See section 3.2 for general information on hypothesis testing using Stata. While often in this book, we show how to conduct both Wald and LR tests of the same hypothesis, in practice you would want to test a hypothesis with only one type of test.

5.3.1 Testing individual coefficients

Most often, we are interested in testing $H_0: \beta_k = 0$, which corresponds to results in column `z` in the output from `logit` and `probit`. For example, consider the results for variables `k5` and `wc` from the `logit` output generated in section 5.2.1:


```
. logit lfp k5 i.wc i.hc k618 i.agecat lwg inc, nolog
Logistic regression               Number of obs   =       753
                                LR chi2(8)         =      124.30
                                Prob > chi2         =       0.0000
Log likelihood = -452.72367       Pseudo R2      =       0.1207
```

lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
k5	-1.391567	.1919279	-7.25	0.000	-1.767739	-1.015395
wc						
college	.7977136	.2291814	3.48	0.001	.3485263	1.246901
(output omitted)						

We conclude the following:

Having young children has a significant effect on the probability of being in the labor force ($z = -7.25$, $p < 0.01$ for a two-tailed test).

The effect of the wife attending college is significant at the 0.01 level.

Testing single coefficients using test

The z test included in the output of estimation commands is a Wald test, which can also be computed as a chi-squared test by using `test`. For example, to test $H_0: \beta_{k5} = 0$,

```
. test k5
( 1) [lfp]k5 = 0
      chi2( 1) =    52.57
      Prob > chi2 =    0.0000
```

Stata refers to the coefficient for `k5` as `[lfp]k5` because the dependent variable is `lfp`. We conclude the following:

The effect of having young children on entering the labor force is significant at the 0.01 level ($\chi^2 = 52.57$, $df = 1$, $p < 0.01$).

The value of the z test is identical to the square root of the corresponding chi-squared test with 1 degree of freedom. For example, using `display` as a calculator,

```
. display sqrt(52.57)
7.2505172
```

This corresponds to -7.25 from the `logit` output shown above.

Aside: Using returns. Using returned results is a better way to show this. When you use the `test` command, the chi-squared statistic is returned as the scalar `r(chi2)`. The command `display sqrt(r(chi2))` then provides the same result more elegantly and with slightly more accuracy.

When using factor variables, such as `i.wc` in our specification of the model, `test` requires that you specify the symbolic name of the coefficient being tested, which for factor variables is not the name used to label the estimate in the output. Specifying either `i.wc` or `wc` in our example will not work with `test`. Instead, the correct command is `test 1.wc`, which indicates that our test is for the coefficient for the indicator variable that `wc` equals 1. Recall that you can list the symbolic names for each coefficient by retyping the name of the estimation command along with the option `coeflegend`, such as `logit, coeflegend`.

Testing single coefficients using `lrtest`

An LR test is computed by comparing the log likelihood from a full model with that of a restricted model. To test a single coefficient, we begin by fitting the full model and storing the results:

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
      (output omitted)
. estimates store Mfull
```

Then, we fit the model without `k5` and store the results:

```
. logit lfp k618 i.agecat i.wc i.hc lwg inc, nolog
      (output omitted)
. estimates store Mnok5
```

Next, we run `lrtest`:

```
. lrtest Mfull Mnok5
Likelihood-ratio test                                LR chi2(1) =      62.55
(Assumption: Mnok5 nested in Mfull)                  Prob > chi2 =    0.0000
```

The LR test shows the following:

The effect of having young children is significant at the 0.01 level ($LR \chi^2 = 62.55$, $df = 1$, $p < 0.01$).

If you want to run an LR test comparing a model stored by using `estimates store` with the last model fit, you can use a single period to represent the last model. For example, instead of `lrtest Mfull Mnok5`, you could use `lrtest Mfull .`, where `.` represents the last model.

5.3.2 Testing multiple coefficients

You might want to test complex hypotheses that involve more than one coefficient. For example, we have two variables that reflect education in the family: `hc` and `wc`. The conclusion that education has (or does not have) a significant effect on labor force participation cannot be based on separate tests of single coefficients. A joint hypothesis can also be tested using either `test` or `lrtest`. Similarly, to test the effect of `agecat` requires testing the coefficients of all indicator variables.

Testing multiple coefficients using `test`

To test that the effect of the wife attending college and of the husband attending college on labor force participation are simultaneously equal to 0 (that is, $H_0: \beta_{wc} = \beta_{hc} = 0$), we fit the full model and test the two coefficients. We must use `1.hc` and `1.wc`, not `hc` and `wc`:

```
. estimates restore Mfull
. test 1.hc 1.wc
( 1)  [lfp]1.hc = 0
( 2)  [lfp]1.wc = 0

      chi2( 2) =    17.83
Prob > chi2 =    0.0001
```

We conclude the following:

We reject the hypothesis that the effects of the husband's and the wife's education are simultaneously equal to 0 ($\chi^2 = 17.83$, $df = 2$, $p < 0.01$).

`test` can also be used to test the equality of coefficients. For example, to test that the effect of the wife attending college on labor force participation is equal to the effect of the husband attending college (that is, $H_0: \beta_{wc} = \beta_{hc}$), we type

```
. test 1.hc = 1.wc
( 1)  - [lfp]1.wc + [lfp]1.hc = 0

      chi2( 1) =     3.24
Prob > chi2 =    0.0719
```

Here `test` translated $\beta_{wc} = \beta_{hc}$ into the equivalent expression $-\beta_{wc} + \beta_{hc} = 0$. The null hypothesis that the effects of husband's and wife's education are equal is marginally significant. We might conclude the following:

There is weak evidence that the effects of husband's and wife's education are equal ($\chi^2 = 3.24$, $df = 1$, $p = 0.072$).

We can test that the effect of `agecat` is 0 by specifying the two indicator variables that were created from the factor variable `i.agecat`:


```
. test 2.agecat 3.agecat
( 1)  [lfp]2.agecat = 0
( 2)  [lfp]3.agecat = 0
      chi2( 2) =    24.27
      Prob > chi2 =    0.0000
```

To avoid having to specify each of the automatically created indicators, we can use `testparm`:

```
. testparm i.agecat
( 1)  [lfp]2.agecat = 0
( 2)  [lfp]3.agecat = 0
      chi2( 2) =    24.27
      Prob > chi2 =    0.0000
```

The advantage of `testparm` is that it works no matter how many indicator variables have been created by `i.catvar`.

Testing multiple coefficients using `lrtest`

To compute an LR test of multiple coefficients, we start by fitting the full model and saving the results with `estimates store estname`. To test the hypothesis that the effect of the wife attending college and of the husband attending college on labor-force participation are both equal to 0 (that is, $H_0: \beta_{wc} = \beta_{hc} = 0$), we fit the model that excludes these two variables and then run `lrtest`:

```
. logit lfp k5 k618 i.agecat lwg inc, nolog
(output omitted)
. estimates store Mnowchc

. lrtest Mfull Mnowchc
Likelihood-ratio test                                LR chi2(2) =    18.68
(Assumption: Mnowchc nested in Mfull)                 Prob > chi2 =    0.0001
```

We conclude the following:

The hypothesis that the effects of the husband's and the wife's education are simultaneously equal to 0 can be rejected at the 0.01 level (LR $\chi^2 = 18.68$, $df = 2$, $p < 0.01$).

This logic can be extended to exclude other variables. Say that we wish to test the hypothesis that the effects of all the independent variables are simultaneously 0. We do not need to fit the full model again because the results are still saved from our use of `estimates store Mfull` above. We fit the model with no independent variables and then run `lrtest`:


```

. logit lfp, nolog
Logistic regression               Number of obs   =       753
                                LR chi2(0)         =        0.00
                                Prob > chi2         =         .
Log likelihood = -514.8732        Pseudo R2       =       0.0000

```

	lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
	_cons	.275298	.0735756	3.74	0.000	.1310925 .4195036

```

. estimates store Mconstant

. lrtest Mfull Mconstant
Likelihood-ratio test               LR chi2(8) =    124.30
(Assumption: Mconstant nested in Mfull) Prob > chi2 =    0.0000

```

We conclude the following:

We reject the hypothesis that all coefficients except the intercept are 0 (LR $\chi^2 = 124.30$, $df = 8$, $p < 0.01$).

This test is identical to the test in the header of the logit output from the full model: LR chi2(8) = 124.30.

5.3.3 Comparing LR and Wald tests

Although the LR and Wald tests are asymptotically equivalent, their values differ in finite samples. In our example,

Hypothesis	df	LR test		Wald test	
		G^2	p	W	p
$\beta_{k5} = 0$	1	62.55	< 0.01	52.57	< 0.01
$\beta_{wc} = \beta_{hc} = 0$	2	18.68	< 0.01	17.83	< 0.01
$\beta_{2.agecat} = \beta_{3.agecat} = 0$	2	25.42	< 0.01	24.27	< 0.01
All slopes = 0	8	124.30	< 0.01	95.90	< 0.01

Statistical theory is unclear on whether the LR or Wald test is to be preferred in models for categorical outcomes, although many statisticians, ourselves included, prefer the LR test. The choice of which test to use is often determined by convenience, personal preference, and convention within an area of research. Recall from chapter 3 that if robust standard errors or svy estimation is used, only Wald tests are available. For Wald tests of a single coefficient, some disciplines prefer to use chi-squared tests, while others prefer the corresponding z test.

5.4 Predicted probabilities, residuals, and influential observations

For a given set of values of the independent variables, the predicted probability can be computed from the estimated coefficients:

$$\text{Logit: } \widehat{\Pr}(y = 1 \mid \mathbf{x}) = \Lambda(\mathbf{x}\hat{\beta}) \quad \text{Probit: } \widehat{\Pr}(y = 1 \mid \mathbf{x}) = \Phi(\mathbf{x}\hat{\beta})$$

where Λ is the CDF for the logistic distribution with variance $\pi^2/3$ and Φ is the CDF for the normal distribution with variance 1. For any set of values of the independent variables, whether occurring in the sample or not, the predicted probability can be computed. In this section, we consider predictions for each observation in the dataset along with residuals and measures of influence based on these predictions. In sections 6.2–6.6, we use predicted probabilities for interpretation.

5.4.1 Predicted probabilities using predict

After running `logit` or `probit`,

```
predict newvar [if] [in]
```

computes the predicted probability of a positive outcome for each observation, given the observed values on the independent variables, and saves them in the new variable *newvar*.

Predictions are computed for all cases in memory that do not have missing values for any variables in the model, regardless of whether `if` and `in` were used to restrict the estimation sample. For example, if you fit `logit lfp k5 i.agecat if wc==1`, only 212 cases are used when fitting the model. But `predict newvar` computes predictions for all 753 cases in the dataset. If you want predictions only for the estimation sample, you can use the command `predict newvar if e(sample)==1`, where `e(sample)` is the variable created by `logit` or `probit` to indicate whether a case was used when fitting the model.

We can use `predict` to examine the range of predicted probabilities from our model. For example, we start by computing the predictions:

```
. predict prlogit
(option pr assumed; Pr(lfp))
```

Because we did not specify which quantity to predict, the default option `pr` for the probability of a positive outcome was assumed, and the new variable `prlogit` was given the default variable label `Pr(lfp)`. In general, and especially when multiple models are being fit, we suggest adding your own variable label to the prediction to avoid having multiple variables with the same label. Here we add a variable label and compute summary statistics:

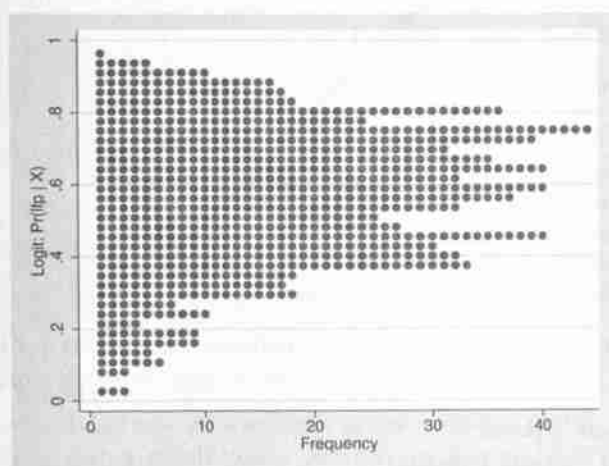

```
. label var prlogit "Logit: Pr(lfp | X)"
```

```
. codebook prlogit, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
prlogit	753	753	.5683931	.0135618	.9512301	Pr(lfp)

The predicted probabilities range from 0.014 to 0.951 with a mean of 0.568. We use `dotplot` to examine the distribution of predictions:⁴

```
. dotplot prlogit, ylabel(0(.2)1, grid gmin gmax)
```



Examining the distribution of predictions is a valuable first step after fitting your model to get a general sense of your data and possibly detect problems. Our plot shows that there are individuals with predicted probabilities that span almost the entire range from 0 to 1, with roughly two-thirds of the observations between 0.40 and 0.80. The large range reflects that our sample contains individuals with both very large and very small probabilities of labor force participation. Examining the characteristics of these individuals could be useful for guiding later analysis. If the distribution was bimodal, it would suggest the importance of a binary independent variable or the possibility of two types of individuals, perhaps with shared characteristics on many variables.

Comparing logit to probit predictions

`predict` can also be used to show that the predictions from logit and probit models are nearly identical. Although the two models make different assumptions about the

4. In this example of `dotplot`, the option `ylabel(0(.2)1, grid gmin gmax)` sets the range of the axis from 0 to 1 with grid lines in increments of 0.2, where the `gmin` and `gmax` suboptions add lines at the minimum and maximum values. Even if the actual range of the predictions is smaller than 0 to 1, we find it useful to see the distribution relative to the entire potential range of probabilities.

distribution of ε , these differences are absorbed in the relative magnitudes of the estimated coefficients. To see this, we begin by fitting comparable logit and probit models and computing their predicted probabilities. First, we fit the logit model, store the estimates, and compute predictions:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. estimates store logit
. predict prlogit
(option pr assumed; Pr(lfp))
. label var prlogit "Logit: Pr(lfp | X)"
```

Next, we fit the probit model:

```
. probit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. estimates store probit
. predict prprobit
(option pr assumed; Pr(lfp))
. label var prprobit "Probit: Pr(lfp | X)"
```

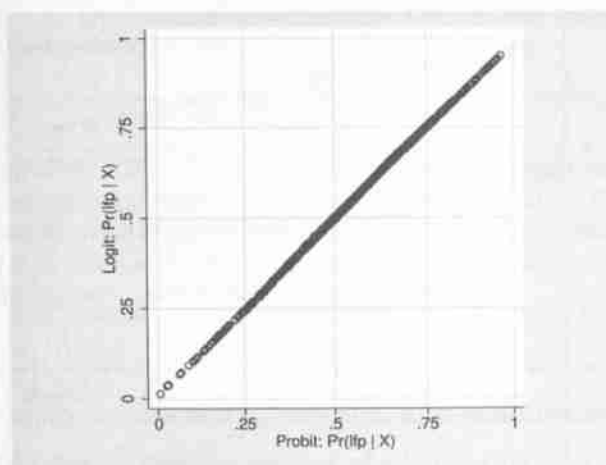
Even though we showed earlier that logit coefficients are about 1.7 times larger than probit coefficients, the predicted probabilities are almost perfectly correlated:

```
. pwcorr prlogit prprobit
```

	prlogit	prprobit
prlogit	1.0000	
prprobit	0.9998	1.0000

Their nearly identical magnitudes can be shown by plotting the predictions from the two models against one another:

```
- scatter prlogit prprobit, xlabel(0(.25)1, grid) ylabel(0(.25)1, grid)
> msymbol(Oh) aspect(1)
```



In terms of predictions, there is little reason to prefer either logit or probit. If your substantive findings rely on whether you used logit or probit, we would not place much confidence in either result unless you have a strong theoretical justification for why one model is preferable to the other. In our own research, we tend to use logit, primarily because we use the multinomial logit model for nominal outcomes. Logit models also allow interpretation in terms of odds ratios, while probit models do not. Given limitations of what can be learned from odds ratios (see section 6.1.1), this alone is not a compelling reason to use the logit model.

5.4.2 Residuals and influential observations using predict

After you have fit your baseline model, we suggest that you examine residuals and look for influential observations before beginning postestimation analyses for interpretation. Residuals and influential observations can help you discover problems with your data and sometimes suggest problems in your model specification. Residuals are the difference between a model's predicted and observed outcomes for each observation in the sample. Cases that have large residuals are known as outliers. When an observation has a large effect on the estimated parameters, it is said to be influential. We illustrate these ideas with the linear regression model in figure 5.3.

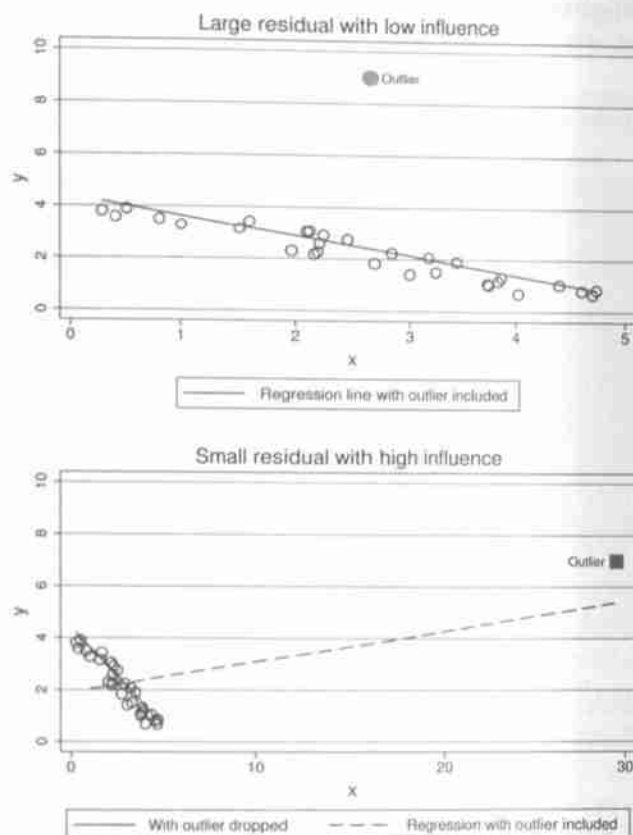


Figure 5.3. The distinction between an outlier and an influential observation

Not all outliers are influential, as the figure shows by using simulated data for the linear regression of y on x . The residual for an observation is its vertical distance from the regression line. In the top panel, the observation highlighted with a solid circle has a large residual and is considered an outlier. Even so, it is not very influential on the slope of the regression line. That is, the slope of the regression line is very close to what it would be if the highlighted observation was dropped from the sample and the model was fit again. In the bottom panel, the only observation whose value has changed is the highlighted observation marked with a square. The residual for this observation is small, but the observation is very influential; its presence is entirely responsible for the slope of the new regression line being positive instead of negative.

Building on the analysis of residuals and influence in the linear regression model (see Fox [2008] and Weisberg [2005, chap. 5]), Pregibon (1981) extended these ideas to the BRM.

Residuals

The predicted probability for a given set of independent variables is

$$\pi_i = \Pr(y_i = 1 \mid \mathbf{x}_i)$$

The deviations $y_i - \pi_i$ are heteroskedastic because the variance depends on the probability π_i of a positive outcome:

$$\text{Var}(y_i - \pi_i \mid \mathbf{x}_i) = \pi_i(1 - \pi_i)$$

The variance is greatest when $\pi_i = 0.5$ and decreases as π_i approaches 0 or 1. That is, a fair coin is the most unpredictable, with a variance of $0.5(1 - 0.5) = 0.25$. A coin that has a very small probability of landing head up (or tail up) has a small variance, for example, $0.01(1 - 0.01) = 0.0099$. The Pearson residual divides the residual $y - \hat{\pi}$ by its standard deviation:

$$r_i = \frac{y_i - \hat{\pi}_i}{\sqrt{\hat{\pi}_i(1 - \hat{\pi}_i)}}$$

Large values of r suggest a failure of the model to fit a given observation.

Pregibon (1981) showed that the variance of r is not 1 because $\text{Var}(y_i - \hat{\pi}_i)$ is not exactly equal to the estimate $\hat{\pi}_i(1 - \hat{\pi}_i)$. He proposed the standardized Pearson residual

$$r_i^{\text{std}} = \frac{r_i}{\sqrt{1 - h_{ii}}}$$

where

$$h_{ii} = \hat{\pi}_i(1 - \hat{\pi}_i) \mathbf{x}_i' \widehat{\text{Var}}(\hat{\boldsymbol{\beta}}) \mathbf{x}_i \quad (5.3)$$

Although r^{std} is preferred over r because of its constant variance, we find that the two residuals are often similar in practice. However, because r^{std} is simple to compute in Stata, you should use this measure.

Example

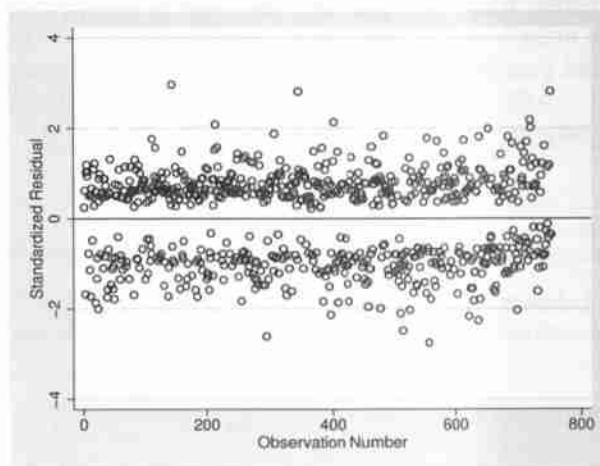
An index plot is an easy way to examine residuals by plotting them against the observation number. Standardized residuals are computed by specifying the `rstandard` option with `predict`. For example,

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
    (output omitted)
. predict rstd, rstandard
. label var rstd "Standardized Residual"
. sort inc
. generate index = _n
. label var index "Observation Number"
```


After computing the standardized residuals that are saved in the new variable `rstd`, we sorted the cases by `inc` so that observations are ordered from lowest to highest income in the plot that follows. The next line creates the variable `index` equal to the observation's row number in the dataset, where `_n` on the right side of `generate` inserts the observation number.

All that remains is to plot the residuals against the index with the following command:

```
. graph twoway scatter rstd index, msymbol(Oh) mcolor(black)
> xlabel(0(200)800) ylabel(-4(2)4, grid gmin gmax) yline(0, lcolor(black))
```



There is no absolute standard that defines a “large” residual. In their discussion of residuals and outliers in the BRM, Hosmer, Lemeshow, and Sturdivant (2013, 193) sagely caution that “in practice, an assessment of ‘large’ is, of necessity, a judgment call based on experience and the particular set of data being analyzed”.

One way to search for problematic residuals is to sort the residuals by the value of a variable that you think may be a problem for the model. Here we sorted on `inc` before plotting. If this variable had been responsible for the lack of fit of some observations, the plot might show a disproportionate number of cases with large residuals among either the low-income or the high-income observations. However, this does not appear to be the case for these data.

Still, several residuals stand out as being large relative to the others. In such cases, it is important to identify those specific observations for further inspection. We can do this by labeling the points with their index number from the variable `index`. We make the marker invisible with `msymbol(none)`, use `mlabel(index)` to specify that we want to label each point with the value contained in variable `index`, and use `mlabposition(0)` to place the label where the marker would have appeared:


```
. sort rstd
. list rstd index k5 agecat wc lwg inc if rstd>1.7 | rstd<-1.7, clean
```

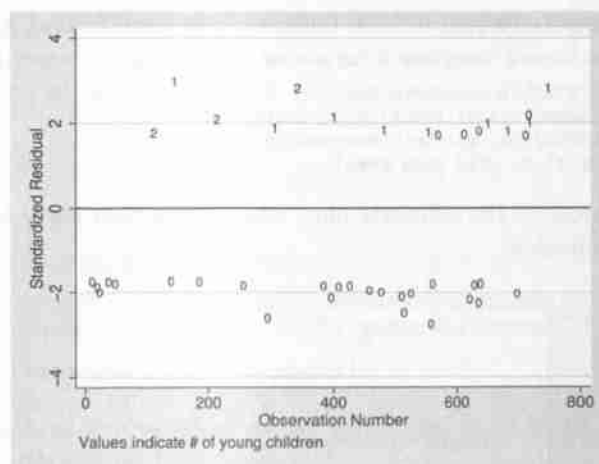
	rstd	index	k5	agecat	wc	lwg	inc
1.	-2.75197	555	0	30-39	college	1.493441	24
2.	-2.605707	297	0	30-39	college	1.065162	15.9
3.	-2.483793	511	0	30-39	college	1.150608	22
4.	-2.258183	637	0	30-39	college	1.432793	28.63
5.	-2.168196	622	0	30-39	college	1.156383	28
6.	-2.129239	396	0	30-39	college	.9602645	18.214
7.	-2.102978	507	0	30-39	college	1.013849	21.85
8.	-2.032079	701	0	30-39	college	1.472049	37.25
9.	-2.025422	522	0	40-49	college	1.526319	22.3

(output omitted)

740.	1.765624	108	2	30-39	no	2.107018	10.2
741.	1.781235	551	1	40-49	no	1.241269	23.709
742.	1.8124	686	1	30-39	no	.9681486	33.97
743.	1.813577	638	0	40-49	no	-.6931472	28.7
744.	1.834293	480	1	40-49	no	.8783384	20.989
745.	1.879891	309	1	30-39	college	-1.543298	16.12
746.	1.988289	653	1	30-39	no	.114816	30.235
747.	2.014942	722	1	30-39	no	.9162908	43
748.	2.084739	214	2	30-39	college	0	13.665
749.	2.138727	401	1	50+	no	1.290984	18.275
750.	2.186398	721	0	50+	no	.6931472	42.91
751.	2.81468	345	2	30-39	no	.5108258	17.09
752.	2.821258	752	1	30-39	college	1.299283	91
753.	2.968983	142	1	30-39	no	-2.054124	11.2

All the cases with the most negative residuals have k5 equal to 0, and cases with positive residuals often have young children. We can also look at this information by modifying the index plot to show only large residuals and to use the option `mlabel(k5)` to label each point with the value of k5:

```
. graph twoway scatter rstd index if (rstd>1.7) | (rstd<-1.7),
> msymbol(none) mlabel(k5) mlabposition(0) xlabel(0(200)800)
> ylabel(-4(2)4, grid gmin gmax) yline(0, lcolor(black))
> caption("Values indicate # of young children")
```

Further analyses of the highlighted cases might reveal either incorrectly coded data or some inadequacy in the specification of the model. If data problems were found, we would correct them. However, cases with large positive or negative residuals should not simply be discarded, because with a correctly specified model you would expect some observations to have large residuals. Instead, you should examine cases with large residuals to see if they suggest a problem with the model (for example, problems associated with the specification of one of the regressors) or errors in the data (for example, a value of 9999 that should have been coded as missing). In our cases above, we found no problems, and coding *k5* as a set of indicator variables did not improve the fit.

Influential cases

As shown in figure 5.3, large residuals do not necessarily have a strong influence on the estimated parameters, and cases with small residuals can have a strong influence. Influential observations, sometimes called high-leverage points, are determined by examining the changes in the estimated β 's that occur when the i th observation is deleted. Although fitting a new logit after eliminating each case is often impractical, Pregibon (1981) derived an approximation that requires fitting the model only once. His delta-beta influence statistic summarizes the effect of removing the i th observation on the entire vector $\hat{\beta}$, which is the counterpart to Cook's distance for the linear regression model. The measure is

$$\Delta \hat{\beta}_i = \frac{r_i^2 h_{ii}}{(1 - h_{ii})^2}$$

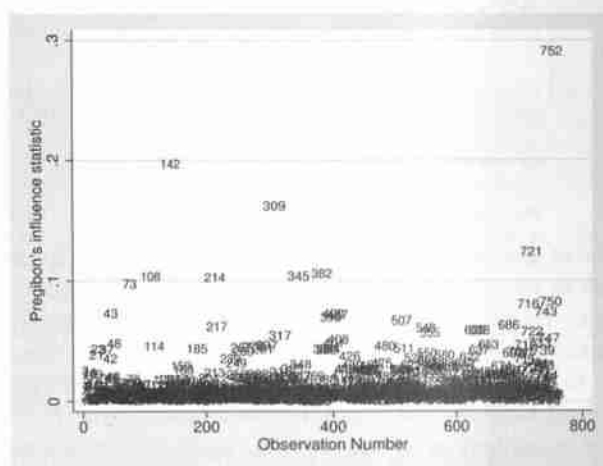
where h_{ii} was defined in (5.3). Stata refers to $\Delta \hat{\beta}_i$ as *dbeta*, which we compute as follows:


```

. predict deltabeta, dbeta
. label var deltabeta "Pregibon's influence statistic"
. graph twoway scatter deltabeta index,
>   msymbol(none) mlabel(index) mlabposition(0)
>   xlabel(0(200)800) xtitle("Observation Number")
>   ylabel(0(.1).3, grid gmin gmax)

```

These commands produce the following plot, which shows that cases 142, 309, and 752 merit further examination:



Methods for plotting residuals and outliers can be extended in many ways, including plots of different diagnostics against one another. Details of these plots can be found in Cook and Weisberg (1999), Hosmer, Lemeshow, and Sturdivant (2013), and Landwehr, Pregibon, and Shoemaker (1984).

5.4.3 Least likely observations

A common motivation for examining residuals in the linear regression model is to uncover the largest residuals and check whether there is a reason why the model fits these observations so poorly. Observations with large residuals are those for which the observed values of the dependent variable are most “surprising”, given the regression coefficients and the values of the independent variables. In this context, we can think of the most surprising outcomes as those that have the smallest predicted probabilities of observing that outcome. These cases may warrant closer inspection, precisely as observations with large residuals do.

The command `leastlikely` (Freese 2002) will list the least likely observations. For example, for a binary model, `leastlikely` will list both the observations with the smallest $\widehat{\Pr}(y = 0 \mid \mathbf{x})$ when $y = 0$ and the observations with smallest $\widehat{\Pr}(y = 1 \mid \mathbf{x})$ when $y = 1$.⁵

Syntax

The syntax for `leastlikely` is as follows:

```
leastlikely [varlist] [if] [in] [, n(#) generate(varname)
           [nodisplay | display] nolabel noobs]
```

where *varlist* contains any variables whose values are to be listed in addition to the observation numbers and probabilities.

Options

`n(#)` specifies the number of observations to be listed for each level of the outcome variable. The default is `n(5)`. For multiple observations with identical predicted probabilities, all observations will be listed.

`generate(varname)` specifies that the probabilities of observing the outcome that was observed be stored in *varname*.

Options controlling the list of values

`leastlikely` can also include any of the options available after `list`. These include the following:

`[no]display` forces the format into display or tabular (`nodisplay`) format. If you do not specify one of these options, Stata chooses the one it decides will be most readable.

`nolabel` causes numeric values rather than labels to be displayed.

`noobs` suppresses printing of the observation numbers.

5. In addition to being used after `logit` and `probit`, `leastlikely` can be used after most binary models in which the option `pr` for `predict` generates the predicted probabilities of a positive outcome (for example, `cloglog`, `scobit`, and `hetprobit`) and after many models for ordinal or nominal outcomes in which the option `outcome(#)` for `predict` generates the predicted probability of outcome `#` (for example, `ologit`, `oprobit`, `mlogit`, `mprobit`, and `slogit`). `leastlikely` is not appropriate for models in which the probabilities produced by `predict` are probabilities within groups or panels (for example, such as `clogit`, `nlogit`, and `asmprobit`).

Example

We can use `leastlikely` to identify the least likely observations from our model of labor force participation and to list the values of the variables `k5`, `k618`, and `wc` for these observations. Based on our model `logit lfp k5 k618 i.agecat i.wc i.hc lwg inc`,

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. leastlikely k5 k618 wc agecat
Outcome: 0 (not in LF)
```

	Prob	k5	k618	wc	agecat
60.	.1403351	0	1	college	30-39
172.	.1291429	0	2	college	30-39
221.	.1765296	0	2	college	30-39
252.	.1174123	0	0	college	30-39
262.	.1650845	0	3	college	30-39

Outcome: 1 (in LF)

	Prob	k5	k618	wc	agecat
427.	.1766887	0	5	no	50+
496.	.1809571	1	0	no	50+
534.	.1039264	1	2	no	30-39
635.	.1152245	1	3	college	30-39
662.	.1133818	2	0	no	30-39

Among women not in the labor force, we find that the lowest predicted probability of not being in the labor force occurs for those who have young children, attended college, and are younger. For women in the labor force with the lowest probabilities of being in the labor force, all but one individual have young children, most have more than one older child, and one attends college. This suggests further consideration of how labor force participation is affected by having children in the family.

5.5 Measures of fit

As discussed in chapter 3, a scalar measure of fit can be useful when comparing competing models. Information criteria such as the Bayesian information criterion (BIC) and Akaike's information criterion (AIC) can be used to select among models and are often very useful. There are many pseudo- R^2 statistics that are inspired by the coefficient of determination R^2 in the linear regression model, but we find them less informative, even though they are often used. Finally, the Hosmer–Lemeshow statistic is a popular way to assess the overall fit of the model, but we do not recommend it for reasons we explain below.

5.5.1 Information criteria

To illustrate scalar measures of fit and information criteria, we consider two models. M_1 contains our original specification of independent variables `k5`, `k618`, `agecat`, `wc`, `hc`, `lw`, and `inc`. M_2 drops variables `k618`, `hc`, and `lw`, and adds income-squared with `c.inc##c.inc`. The models are fit and estimates are stored:

```
. quietly logit lfp k5 k618 i.agecat i.wc i.hc lw inc, nolog
. estimates store model1
. quietly logit lfp k5 i.agecat i.wc c.inc##c.inc, nolog
. estimates store model2
```

We can list the estimates by using the option `stats(N bic aic r2_p)` to include the sample size, BIC, AIC, and pseudo- R^2 that is normally included in models fit with maximum likelihood. Recall that the formulas for these statistics are given in section 3.3.2.

```
. estimates table model1 model2, b(%9.3f) p(%9.3f) stats(N bic aic r2_p)
```

Variable	model1	model2
k5	-1.392	-1.369
	0.000	0.000
k618	-0.066	
	0.336	
agecat		
40-49	-0.627	-0.512
	0.003	0.011
50+	-1.279	-1.137
	0.000	0.000
wc		
college	0.798	1.119
	0.001	0.000
hc		
1	0.136	
	0.508	
lw	0.610	
	0.000	
inc	-0.035	-0.060
	0.000	0.001
c.inc#c.inc		0.000
		0.083
_cons	1.014	1.743
	0.000	0.000
N	753	753
bic	965.064	968.574
aic	923.447	936.206
r2_p	0.121	0.104

legend: b/p

M_2 modifies M_1 by deleting a statistically significant variable and two nonsignificant variables from M_1 while adding a variable that is significant at the 0.10 level. Because the models are not nested, they cannot be compared with an LR test, but we can use the BIC and AIC statistics. In this example, both the BIC and AIC statistics are smaller for M_1 , which provides support for that model. Following Raftery's (1995) guidelines, we can say that there is positive (neither weak nor strong) support for M_1 .

You can obtain information criteria in two other ways. You can use the `ic` option to `fitstat`, which shows multiple versions of the AIC and BIC measures (see section 3.3 for the formula for these measures):

```
. estimates restore model1
(results model1 are active now)
. quietly fitstat, save ic
. estimates restore model2
(results model2 are active now)
. fitstat, diff ic
```

		Current	Saved	Difference
AIC				
	AIC	936.206	923.447	12.759
	(divided by N)	1.243	1.226	0.017
BIC				
	BIC (df=7/9/-2)	968.574	965.064	3.511
	BIC (based on deviance)	-4019.347	-4022.857	3.511
	BIC* (based on LRX2)	-67.796	-71.307	3.511

Difference of 3.511 in BIC provides positive support for saved model.

The results match those from the `estimates` table output and even tell you the strength of support for the preferred model. You can also use the `estat ic` command:

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
```

```
. estat ic
```

Akaike's information criterion and Bayesian information criterion

Model	Obs	ll(null)	ll(model)	df	AIC	BIC
.	753	-514.8732	-452.7237	9	923.4473	965.0639

Note: N=Obs used in calculating BIC; see [R] BIC note

```
. logit lfp k5 i.agecat i.wc c.inc##c.inc, nolog
(output omitted)
```



```
. estat ic
```

Akaike's information criterion and Bayesian information criterion

Model	Obs	ll(null)	ll(model)	df	AIC	BIC
.	753	-514.8732	-461.103	7	936.206	968.5745

Note: N=Obs used in calculating BIC; see [R] BIC note

These results match those from `fitstat`.

5.5.2 Pseudo- R^2 's

Within a substantive area, pseudo- R^2 's might provide a rough index of whether a model is adequate. For example, if prior models of labor force participation routinely have values of 0.4 for a particular pseudo- R^2 , you would expect that new analyses with a different sample or with revised measures of the variables would result in a similar value for that measure. But there is no convincing evidence that selecting a model that maximizes the value of a pseudo- R^2 results in a model that is optimal in any sense other than the model has a larger value of that measure.

We use the same models estimated in the last section and use `fitstat` to compute the scalar measures of fit (see section 3.3 for the formula for these measures):

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. quietly fitstat, save
```



```
. logit lfp k5 i.agecat i.wc c.inc##c.inc, nolog
(output omitted)
. fitstat, diff
```

	Current	Saved	Difference
Log-likelihood			
Model	-461.103	-452.724	-8.379
Intercept-only	-514.873	-514.873	0.000
Chi-square			
D (df=746/744/2)	922.206	905.447	16.759
LR (df=6/8/-2)	107.540	124.299	-16.759
p-value	0.000	0.000	0.000
R2			
McFadden	0.104	0.121	-0.016
McFadden (adjusted)	0.091	0.103	-0.012
McKelvey & Zavoina	0.183	0.215	-0.032
Cox-Snell/ML	0.133	0.152	-0.019
Cragg-Uhler/Nagelkerke	0.179	0.204	-0.026
Efron	0.135	0.153	-0.018
Tjur's D	0.135	0.153	-0.018
Count	0.672	0.676	-0.004
Count (adjusted)	0.240	0.249	-0.009
IC			
AIC	936.206	923.447	12.759
AIC divided by N	1.243	1.226	0.017
BIC (df=7/9/-2)	968.574	965.064	3.511
Variance of			
e	3.290	3.290	0.000
y-star	4.026	4.192	-0.165

Note: Likelihood-ratio test assumes current model nested in saved model.

Difference of 3.511 in BIC provides positive support for saved model.

After fitting our first model with `logit`, we used `quietly` to suppress the output from `fitstat` with the `save` option to retain the results in memory. After fitting the second model, `fitstat, diff` displays the fit statistics for both models. `fitstat, diff` computes differences between all measures, shown in the column labeled *Difference*, even if the models are not nested. As with the `lrtest` command, you must determine if it makes sense to interpret the computed difference.

What do the fit statistics show? The values of the pseudo- R^2 's are slightly larger for M_1 , which is labeled *Saved logit* in the table. If you take the pseudo- R^2 's as evidence for the best model, which we do not, there is some evidence preferring M_1 .

5.5.3 (Advanced) Hosmer–Lemeshow statistic

We only recommend reading this section if you are considering using the Hosmer–Lemeshow statistic. After reviewing how the measure is computed, we illustrate that the statistic is highly dependent upon an arbitrary decision on the number of groups used. As a result, we do not recommend this measure.

The idea of the Hosmer–Lemeshow (HL) test is to compare predicted probabilities with the observed data (Hosmer and Lemeshow 1980; Lemeshow and Hosmer 1982). This popular test can be computed using the `estat gof` command after fitting a logit or probit model. Unlike the measures presented above, this command also works with models fit by using complex survey data with the `svy` prefix.

To explain how the test works, we review the steps that are used to compute it.

1. Fit the regression model.
2. Compute the predicted probabilities $\hat{\pi}_i$.
3. Sort the data from the smallest $\hat{\pi}_i$ to the largest.
4. Divide the observations into G groups, where 10 groups are often used. Each group will have $n_g \approx \frac{N}{G}$ cases. The first group will have the n_1 smallest values of $\hat{\pi}_i$, and so on. If G does not divide equally into N , the group sizes will differ slightly.
5. Within each group, compute the mean prediction:

$$\bar{\pi}_g = \sum_{i \text{ in group } g} \hat{\pi}_i / n_g$$

Also compute the mean number of observed cases where $y = 1$:

$$\bar{y}_g = \sum_{i \text{ in group } g} y_i / n_g$$

6. The test statistic is

$$HL = \sum_{g=1}^G \frac{(n_g \bar{y}_g - n_g \bar{\pi}_g)^2}{n_g \bar{\pi}_g (1 - \bar{\pi}_g)}$$

Hosmer, Lemeshow, and Sturdivant (2013) ran simulations that suggest that HL is distributed approximately as χ^2 with $G - 2$ degrees of freedom if the model is correctly specified. If the p -value is large, the model is considered to fit the data.

To give an example, we fit the model we have used as a running example, and we use `estat gof` to compute the HL statistic:


```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg c.inc, nolog
(output omitted)
. estat gof, group(10)
```

Logistic model for lfp, goodness-of-fit test

(Table collapsed on quantiles of estimated probabilities)

```
number of observations =      753
number of groups =      10
Hosmer-Lemeshow chi2(8) =     13.76
Prob > chi2 =      0.0881
```

The p -value is greater than 0.05, indicating that the model fits based on the criterion provided for the HL.

Unfortunately, we do not find conclusions from the HL test to be convincing. First, as Hosmer and Lemeshow point out, the HL test is not a substitute for examining individual predictions and residuals as discussed in the last section. Second, Allison (2012b, 67) raised concerns that the test is not very powerful. In a simple simulation with 500 cases, the HL test failed to reject an incorrect model 75% of the time. Third, and most critically, the choice of the number of groups is arbitrary, even though 10 is most often used. The results of the Hosmer–Lemeshow test are sensitive to the arbitrary choice of the number of groups. In our experience, this is often the case and for this reason we do not recommend the test.

We can illustrate the sensitivity of the Hosmer–Lemeshow test by varying the number of groups used to compute the test from 5 to 15 in the model fit above:

	chi2	df	prob
5 groups	4.043	3.000	0.257
6 groups	8.762	4.000	0.067
7 groups	10.424	5.000	0.064
8 groups	13.831	6.000	0.032
9 groups	15.503	7.000	0.030
10 groups	13.763	8.000	0.088
11 groups	17.980	9.000	0.035
12 groups	24.055	10.000	0.007
13 groups	15.230	11.000	0.172
14 groups	19.360	12.000	0.080
15 groups	24.722	13.000	0.025

The row labeled 10 groups corresponds to the result shown earlier. It is disconcerting that when 10 groups are used, the result is not significant, but if 9 or 11 groups had been used, the result would have been significant. Over the 11 groups listed, the p -values range from $p = 0.007$ to $p = 0.257$. Although the idea of the HL test is appealing, we are skeptical that it is an effective way to assess a model.

5.6 Other commands for binary outcomes

Logit and probit models are the most commonly used models for binary outcomes and are the only ones that we consider in this book, but other models exist that can be fit in Stata. Among them are the following:

- **cloglog** assumes a complementary log-log distribution for the errors instead of a logistic or normal distribution.
- **scobit** fits a logit model that relaxes the assumption that the marginal change in the probability is greatest when $\Pr(y = 1) = 0.5$.
- **hetprobit** allows the assumed variance of the errors in the probit model to vary as a function of the independent variables, which is one approach to comparing logit and probit coefficients across groups (Williams 2009).
- **ivprobit** fits a probit model where one or more of the regressors are endogenously determined.
- **biprobit** simultaneously fits two binary probits and can be used when errors are correlated with each other as in the estimation of seemingly unrelated regression models for continuous dependent variables. **mvprobit** (Cappellari and Jenkins 2003) extends this idea to more than two binary probits.

Binary outcomes that reflect an event that is expected to happen eventually for all cases are often handled using survival analysis, which is not covered in this book. Cleves et al. (2010) provides a detailed introduction to survival analysis in Stata focusing on the **st*** commands. Likewise, we do not consider Stata's extensive commands for working with panel and multilevel data, including Stata's **xt*** and **me*** commands, but these are discussed extensively in Rabe-Hesketh and Skrondal (2012) and Cameron and Trivedi (2010).

5.7 Conclusion

Binary outcomes are at least as common as continuous outcomes in many fields, and the logic for how binary outcomes are handled provides the basis for ordinal, nominal, and count outcomes in later chapters. We focus here on the logit and probit models that are the most common models used for binary outcomes. In this chapter, we considered estimating parameters for these models, testing basic hypotheses about estimates, and evaluating the fit of individual observations and the model as a whole.

We have said nothing so far about how to think and talk about what the estimates produced by these models actually mean. Instead, this serves as the topic of the next chapter. As you will see, the next chapter is longer than this chapter, because there are many issues to think about to figure out the clearest and most effective way to convey results from models for binary outcomes.

1. The first part of the paper discusses the importance of the study.

2. The second part of the paper discusses the methodology used in the study.

3. The third part of the paper discusses the results of the study.

4. The fourth part of the paper discusses the conclusions of the study.

5. The fifth part of the paper discusses the implications of the study.

6. The sixth part of the paper discusses the limitations of the study.

7. The seventh part of the paper discusses the future research.

8. The eighth part of the paper discusses the acknowledgments.

9. The ninth part of the paper discusses the references.

10. The tenth part of the paper discusses the appendices.

11. The eleventh part of the paper discusses the index.

12. The twelfth part of the paper discusses the glossary.

DATA COLLECTION

The data were collected from a random sample of 1,000 households in the United States. The sample was selected using a multi-stage sampling procedure. First, a list of all households in the United States was obtained from the Census Bureau. Then, a random sample of 100 counties was selected. Finally, a random sample of 1,000 households was selected from the 100 counties.

The data were collected using a computer-assisted telephone interview (CATI) system. The CATI system was designed to collect data on a wide range of topics, including household characteristics, income, and health status. The CATI system was used to collect data on a sample of 1,000 households in the United States.

The data were collected from a random sample of 1,000 households in the United States. The sample was selected using a multi-stage sampling procedure. First, a list of all households in the United States was obtained from the Census Bureau. Then, a random sample of 100 counties was selected. Finally, a random sample of 1,000 households was selected from the 100 counties.

The data were collected using a computer-assisted telephone interview (CATI) system. The CATI system was designed to collect data on a wide range of topics, including household characteristics, income, and health status. The CATI system was used to collect data on a sample of 1,000 households in the United States.

The data were collected from a random sample of 1,000 households in the United States. The sample was selected using a multi-stage sampling procedure. First, a list of all households in the United States was obtained from the Census Bureau. Then, a random sample of 100 counties was selected. Finally, a random sample of 1,000 households was selected from the 100 counties.

The data were collected using a computer-assisted telephone interview (CATI) system. The CATI system was designed to collect data on a wide range of topics, including household characteristics, income, and health status. The CATI system was used to collect data on a sample of 1,000 households in the United States.

The data were collected from a random sample of 1,000 households in the United States. The sample was selected using a multi-stage sampling procedure. First, a list of all households in the United States was obtained from the Census Bureau. Then, a random sample of 100 counties was selected. Finally, a random sample of 1,000 households was selected from the 100 counties.

The data were collected using a computer-assisted telephone interview (CATI) system. The CATI system was designed to collect data on a wide range of topics, including household characteristics, income, and health status. The CATI system was used to collect data on a sample of 1,000 households in the United States.

The data were collected from a random sample of 1,000 households in the United States. The sample was selected using a multi-stage sampling procedure. First, a list of all households in the United States was obtained from the Census Bureau. Then, a random sample of 100 counties was selected. Finally, a random sample of 1,000 households was selected from the 100 counties.

The data were collected using a computer-assisted telephone interview (CATI) system. The CATI system was designed to collect data on a wide range of topics, including household characteristics, income, and health status. The CATI system was used to collect data on a sample of 1,000 households in the United States.

The data were collected from a random sample of 1,000 households in the United States. The sample was selected using a multi-stage sampling procedure. First, a list of all households in the United States was obtained from the Census Bureau. Then, a random sample of 100 counties was selected. Finally, a random sample of 1,000 households was selected from the 100 counties.

The data were collected using a computer-assisted telephone interview (CATI) system. The CATI system was designed to collect data on a wide range of topics, including household characteristics, income, and health status. The CATI system was used to collect data on a sample of 1,000 households in the United States.

6 Models for binary outcomes: Interpretation

In this chapter, we discuss methods for interpreting results from models for binary outcomes. Because the binary regression model (BRM) that we introduced in the last chapter is nonlinear, the magnitude of the change in the outcome probability associated with a given change in an independent variable depends on the levels of all the independent variables. The challenge of interpreting results, then, is to find a summary of how changes in the independent variables are associated with changes in the outcome that best reflects critical substantive processes without overwhelming yourself or your readers with distracting detail.

Two basic approaches to interpretation are considered: interpretation using regression coefficients and interpretation using predicted probabilities. With respect to the former, we begin the chapter by considering how estimated parameters or simple functions of these parameters can be used to predict changes in the odds ratios for the logit model or the latent variable y^* for logit or probit models. Using odds ratios to interpret the logit model is very common, but rarely is it sufficient for understanding the results of the model. Nonetheless, it is important to understand what odds ratios mean for several reasons. For one, odds ratios are used a lot, and you need to understand what they can and cannot tell you. Also, odds ratios are useful for understanding the structure of the ordinal regression model in chapter 7 and the multinomial logit model in chapter 8. Interpretation based on y^* parallels interpretation in the linear regression model, but it is not often used for binary outcomes. It is, however, sometimes useful for models for ordinal outcomes, considered in chapter 7.

We strongly prefer methods of interpretation that are based on predicted probabilities, and most of the chapter focuses on these. We begin in section 6.2 with marginal effects, which we find more informative than the more commonly used odds ratios as scalar measures to assess the magnitude of a variable's effect. In section 6.3, we consider computing predictions based on substantively motivated profiles of values for the independent variables, also referred to as ideal types. Thinking about the types of individuals represented in your sample is a valuable way to gain an intuitive sense of which configurations of variables are substantively important. Tables of predictions, which are discussed in section 6.4, can effectively highlight the impact of categorical independent variables. We end our discussion of interpretation in section 6.6 by considering graphical methods to show how probabilities change as a continuous independent variable changes.

When using logit or probit, or any nonlinear model, we suggest that you try a variety of methods of interpretation with the goal of finding an elegant way to present the results that does justice to the complexities of the nonlinear model and the substantive application. No one method works in all situations, and often the only way to determine which method is most effective is to try them all. Fortunately, the methods we consider in this chapter can be readily extended to models for ordinal, nominal, and count outcomes, which are considered in chapters 7–9.

6.1 Interpretation using regression coefficients

Interpretation of regression models involves examining how a change in an independent variable is associated with a change in the outcome. In linear models, this is easily accomplished using estimates of the regression coefficients. For example, the coefficient for years of education on anticipated earnings tells you how one more year of education is expected to affect earnings. Unless you add interaction terms, the estimated effect is the same for men and women, whites and nonwhites, and all combinations of values of the independent variables. Consequently, in linear regression, a discussion of the slope coefficients is often where a researcher begins and ends his or her interpretation of the model. The presentation of results might simply be a table of regression coefficients.

In the nonlinear BRM, a regression coefficient indicates the direction of a variable's effect. In our model of labor force participation, the coefficients for $k5$ and $k618$ are both negative, which implies that higher numbers of children are associated with a lower probability of being in the labor force. What is harder to interpret from the coefficient is the magnitude of the effect. The logit model, for example, can be written as

$$\ln \Omega(x) = x\beta$$

The β coefficients indicate the effect of the independent variable on the log odds of the outcome, where the log odds is also known as the logit. We can interpret the β 's as follows:

For a unit change in x_k , we expect the log of the odds of the outcome to change by β_k units, holding all other variables constant.

This interpretation does not depend on the level of x_k or the levels of the other variables in the model. In this regard, it is just like the linear regression model. The problem is that a change of β_k in the log odds has little substantive meaning for most people. Consequently, tables of logit coefficients typically have little value for conveying the magnitude of effects. As an alternative, odds ratios can be used to explain the effects of independent variables on the odds, which we consider in the next section.

6.1.1 Interpretation using odds ratios

Effects for the logit model (but not the probit model) can be interpreted in terms of changes in the odds. For binary outcomes, we typically consider the odds of observing a positive outcome, coded 1, versus a negative outcome, coded 0:

$$\Omega(\mathbf{x}) = \frac{\Pr(y = 1 | \mathbf{x})}{\Pr(y = 0 | \mathbf{x})} = \frac{\Pr(y = 1 | \mathbf{x})}{1 - \Pr(y = 1 | \mathbf{x})}$$

In the logit model, the log odds are a linear combination of the x 's and β 's. For example, consider a model with three independent variables:

$$\ln \left\{ \frac{\Pr(y = 1 | \mathbf{x})}{\Pr(y = 0 | \mathbf{x})} \right\} = \ln \Omega(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

The problem with interpreting these β 's directly is that changes in log odds are not substantively meaningful to most audiences.

To make the interpretation more meaningful, we can transform the log odds to the odds by taking the exponential of both sides of the equation. This leads to a model that is multiplicative instead of linear but in which the outcome is the odds:

$$\Omega(\mathbf{x}, x_3) = e^{\beta_0} e^{\beta_1 x_1} e^{\beta_2 x_2} e^{\beta_3 x_3}$$

Our notation emphasizes the value of x_3 , which we want to increase by 1:

$$\begin{aligned} \Omega(\mathbf{x}, x_3 + 1) &= e^{\beta_0} e^{\beta_1 x_1} e^{\beta_2 x_2} e^{\beta_3 (x_3 + 1)} \\ &= e^{\beta_0} e^{\beta_1 x_1} e^{\beta_2 x_2} e^{\beta_3 x_3} e^{\beta_3} \end{aligned}$$

This leads to the odds ratio

$$\frac{\Omega(\mathbf{x}, x_3 + 1)}{\Omega(\mathbf{x}, x_3)} = \frac{e^{\beta_0} e^{\beta_1 x_1} e^{\beta_2 x_2} e^{\beta_3 x_3} e^{\beta_3}}{e^{\beta_0} e^{\beta_1 x_1} e^{\beta_2 x_2} e^{\beta_3 x_3}} = e^{\beta_3} \quad (6.1)$$

Accordingly, we can interpret the exponential of the logit coefficient as follows:

For a unit change in x_k , the odds are expected to change by a factor of $\exp(\beta_k)$, holding other variables constant.

For $\exp(\beta_k) > 1$, you could say that the odds are " $\exp(\beta_k)$ times larger"; for $\exp(\beta_k) < 1$, you could say that the odds are " $\exp(\beta_k)$ times smaller". If $\exp(\beta_k) = 1$, then x_k does not affect the odds. We can evaluate the effect of a standard deviation change in x_k instead of a unit change:

For a standard deviation change in x_k , the odds are expected to change by a factor of $\exp(\beta_k \times s_k)$, holding all other variables constant.

The odds ratio is computed by changing one variable, while holding all other variables constant. This means that the formula in (6.1) cannot be used when the variable being changed is mathematically linked to another variable. For example, if x_1 is age and x_2 is age-squared, you cannot increase x_1 by 1 while holding x_2 constant. In such cases, the odds ratio computed as $\exp(\beta_k)$ should not be interpreted.

Although odds ratios are a common method of interpretation for logit models, it is essential to understand their limitations. Most importantly, they do not indicate the magnitude of the change in the probability of the outcome. We begin with an example from our model of labor force participation, followed by a few words of caution.

The output from `logit` with the `or` option shows the odds ratios instead of the estimated β 's:

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, or nolog
Logistic regression
```

Number of obs	=	753
LR chi2(8)	=	124.30
Prob > chi2	=	0.0000
Pseudo R2	=	0.1207

```
Log likelihood = -452.72367
```

lfp	Odds Ratio	Std. Err.	z	P> z	[95% Conf. Interval]
k5	.2486853	.0477297	-7.25	0.000	.1707185 .3622592
k618	.9364419	.0639721	-0.96	0.336	.8190905 1.070606
agecat					
40-49	.5343201	.1115249	-3.00	0.003	.3549247 .8043904
50+	.2782939	.0722959	-4.92	0.000	.1672539 .4630534
wc					
college	2.220458	.5088877	3.48	0.001	1.416978 3.479543
hc					
college	1.145555	.2353502	0.66	0.508	.7658431 1.713532
lwg	1.840265	.2775073	4.04	0.000	1.369372 2.473087
inc	.9655531	.0079868	-4.24	0.000	.9500254 .9813346
_cons	2.756603	.7885231	3.54	0.000	1.573581 4.829025

Here are some examples of interpretations:

For each additional young child, the odds of being in the labor force decrease by a factor of 0.25, holding all other variables constant ($p < 0.01$).

If a woman attended college, her odds of labor force participation are 2.22 times larger than a woman who did not attend college, holding all other variables constant ($p < 0.01$).

Notice that these interpretations contain no information about the magnitude of the implied change in the probability. This will be important to our discussion below of the limitations of odds ratios, but first, several other issues about the interpretation of odds ratios merit attention.

Odds ratios for categorical variables. Multiple odds ratios are associated with a multiple-category independent variable. For example, the two coefficients for `agecat` are relative to the base category of being 30 to 39. Accordingly, we can interpret the coefficient labeled 40-49 as follows:

Being age 40 to 49 compared with being 30 to 39 decreases the odds of being in the labor force by a factor of 0.53, holding all other variables constant ($p < 0.01$).

And similarly for the coefficient for 50+. If you are interested in the coefficient for the effects of being 40 to 49 compared with being 50 or older, you can use the `pwcompare` command, where option `eform` requests the exponential of the coefficients, which are the odds ratios. The option `effects` requests p -values and confidence intervals.

```
. pwcompare agecat, effects eform
```

```
Pairwise comparisons of marginal linear predictions
```

```
Margins      : asbalanced
```

		exp(b)	Std. Err.	Unadjusted z	P> z	Unadjusted [95% Conf. Interval]
lfp						
	agecat					
	40-49					
	vs					
	30-39	.5343201	.1115249	-3.00	0.003	.3549247 .8043904
50+ vs	30-39	.2782939	.0722959	-4.92	0.000	.1672539 .4630534
50+ vs	40-49	.5208373	.1141603	-2.98	0.003	.338946 .8003385

We conclude the following:

Being 50 or older compared with being 40 to 49 decreases the odds of being in the labor force by a factor of 0.52, holding all other variables constant ($p < 0.01$).

Although any two of the pairwise coefficients imply the third (for example, $0.521 \times 0.534 = 0.278$), it is often useful to see all the coefficients and report those that are most useful for the substantive application.

Confidence intervals for odds ratios. If you report the odds ratios instead of the untransformed β coefficients, then the 95% confidence interval of the odds ratio is often reported instead of the standard error. The reason is that the odds ratio is a nonlinear transformation of the logit coefficient, so the confidence interval is asymmetric. For example, if the logit coefficient is 0.75 with a standard error of 0.25, the 95% interval around the logit coefficient is approximately $[0.26, 1.24]$, but the confidence interval around the odds ratio $\exp(0.75) = 2.12$ is $[\exp(0.26), \exp(1.24)] = [1.30, 3.46]$. The `or` option for the `logit` command reports odds ratios and includes confidence intervals.

Odds ratios for changes other than one. You can compute the odds ratio for changes in x_k other than 1 with the formula

$$\frac{\Omega(\mathbf{x}, x_k + \delta)}{\Omega(\mathbf{x}, x_k)} = e^{\delta\beta_k} = e^{\delta} e^{\beta_k}$$

For example,

Increasing income by \$10,000 decreases the odds by a factor of 0.70 ($=e^{-0.035 \times 10}$), holding all other variables constant.

Accordingly, the odds ratio for a standard deviation change of an independent variable equals $\exp(\beta_k s_k)$, where s_k is the standard deviation of x_k . The odds ratios for both a unit and a standard deviation change of the independent variables can be obtained with `listcoef`:

```
. listcoef, help
logit (N=753): Factor change in odds
Odds of: in LF vs not in LF
```

	b	z	P> z	e ^b	e ^b StdX	SDofX
k5	-1.3916	-7.250	0.000	0.249	0.482	0.524
k618	-0.0657	-0.961	0.336	0.936	0.917	1.320
agecat						
40-49	-0.6268	-3.003	0.003	0.534	0.737	0.487
50+	-1.2791	-4.924	0.000	0.278	0.589	0.414
wc						
college	0.7977	3.481	0.001	2.220	1.432	0.450
hc						
college	0.1359	0.661	0.508	1.146	1.069	0.488
lwg	0.6099	4.045	0.000	1.840	1.431	0.588
inc	-0.0351	-4.238	0.000	0.966	0.665	11.635
constant	1.0140	3.545	0.000	.	.	.

```
b = raw coefficient
z = z-score for test of b=0
P>|z| = p-value for z-test
eb = exp(b) = factor change in odds for unit increase in X
ebStdX = exp(b*SD of X) = change in odds for SD increase in X
SDofX = standard deviation of X
```

By using the coefficients for `lwg` in the column `ebStdX`, we can say

For a standard deviation increase in the log of the wife's expected wages, the odds of being in the labor force are 1.43 times greater, holding all other variables constant.

Odds ratios as multiplicative coefficients. When interpreting odds ratios, remember that they are multiplicative. This means that 1 indicates no effect, positive effects are greater than 1, and negative effects are between 0 and 1. Magnitudes of positive and negative effects should be compared by taking the inverse of the negative effect, or vice versa. For example, an odds ratio of 2 has the same magnitude as an odds ratio of $0.5 = 1/2$. Thus an odds ratio of $0.1 = 1/10$ is much "larger" than the odds ratio $2 = 1/0.5$. Another consequence of the multiplicative scale is that to determine the effect on the odds of the event not occurring, you simply take the inverse of the effect on the odds of the event occurring. `listcoef` will automatically calculate this for you if you specify the `reverse` option:

```
. listcoef, reverse
logit (N=753): Factor change in odds
Odds of: not in LF vs in LF
```

	b	z	P> z	e ^b	e ^b StdX	SDofX
k5	-1.3916	-7.250	0.000	4.021	2.073	0.524
k618	-0.0657	-0.961	0.336	1.068	1.091	1.320
agecat						
40-49	-0.6268	-3.003	0.003	1.872	1.357	0.487
50+	-1.2791	-4.924	0.000	3.593	1.698	0.414
wc						
college	0.7977	3.481	0.001	0.450	0.698	0.450
hc						
college	0.1359	0.661	0.508	0.873	0.936	0.488
lwg	0.6099	4.045	0.000	0.543	0.699	0.588
inc	-0.0351	-4.238	0.000	1.036	1.504	11.635
constant	1.0140	3.545	0.000	.	.	.

The header indicates that columns `eb` and `ebStdX` now contain the factor changes in the odds of the outcome `not in LF` versus the outcome `in LF`, whereas before we computed the factor change in the odds of `in LF` versus `not in LF`. We can interpret the result for `k5` as follows:

For each additional child, the odds of not being in the labor force are increased by a factor of 4.02 ($=1/0.48$), holding all other variables constant.

Percentage change in the odds. Instead of a multiplicative or factor change in the outcome, some people prefer the percentage change:

$$\text{percentage change in odds} = 100 \{ \exp(\delta\beta_k) - 1 \}$$

This is shown by `listcoef` with the `percent` option.


```
. listcoef, help percent
logit (N=753): Percentage change in odds
Odds of: in LF vs not in LF
```

	b	z	P> z	%	%StdX	SDofX
k5	-1.3916	-7.250	0.000	-75.1	-51.8	0.524
k618	-0.0657	-0.961	0.336	-6.4	-8.3	1.320
agecat						
40-49	-0.6268	-3.003	0.003	-46.6	-26.3	0.487
50+	-1.2791	-4.924	0.000	-72.2	-41.1	0.414
wc						
college	0.7977	3.481	0.001	122.0	43.2	0.450
hc						
college	0.1359	0.661	0.508	14.6	6.9	0.488
lwg	0.6099	4.045	0.000	84.0	43.1	0.588
inc	-0.0351	-4.238	0.000	-3.4	-33.5	11.635
constant	1.0140	3.545	0.000	.	.	.

```
b = raw coefficient
z = z-score for test of b=0
P>|z| = p-value for z-test
% = percent change in odds for unit increase in X
%StdX = percent change in odds for SD increase in X
SDofX = standard deviation of X
```

Some interpretations are as follows:

For each additional young child, the odds of being in the labor force decrease by 75%, holding all other variables constant.

A standard deviation increase in the log of a wife's expected wages increases the odds of being in the labor force by 43%, holding all other variables constant.

Percentage and factor change provide the same information, so which you use is a matter of preference. Although we tend to prefer percentage change, methods for the graphical interpretation of the multinomial logit model (chapter 8) work only with factor change coefficients.

Limitations of the odds ratio

The interpretation of the odds ratio assumes that the other variables are held constant, but it does not require that they be held at specific values. This might seem to resolve the problem of nonlinearity, but in practice it does not. A constant factor change in the odds does not imply a constant change in the probability, and probabilities provide a more meaningful metric for interpretation than do odds. For example, if the odds are 1/50, the corresponding probability is 0.020 because $p = \Omega/(1 + \Omega)$. If the odds

double to 2/50, the probability increases to 0.038 for a change of 0.019. Depending on your substantive purposes, this small change may be trivial or may be quite important (such as when you identify a risk factor that makes it twice as likely that a subject will contract a fatal disease). Meanwhile, if the odds are 1/1 and double to 2/1, the probability increases 0.500 to 0.667 for a change of 0.167. The odds ratio is the same, but the change in the probability is much larger.

The substantive meaning of a given odds ratio depends on the specific value of the odds before they change. Those odds in turn depend on the predicted probability, which in turn depends on the specific values of all independent variables in the model. For describing results in terms of probabilities, there is no way around the nonlinearity of the model.

Odds ratios in case-control studies

Odds ratios may provide the best alternative for contexts in which the probabilities are determined by the sample design and accordingly are not obviously of substantive interest. The key example is case-control studies, which are especially common in epidemiology. Case-control studies recruit cases with a disease separately from the controls without the disease, and commonly studies will recruit equal proportions of cases and controls even though cases are relatively rare in the population and controls are abundant. In other words, the proportion of people with a disease in a case-control study has nothing to do with the proportion who actually have the disease in a population.

Logit models are typically used in such studies because, if assumptions are satisfied, odds ratios estimated from a case-control study can be extrapolated to a population (Hosmer, Lemeshow, and Sturdivant 2013). However, because the baseline probability in the population is much lower than the proportion in the sample (and might not even be known), interpreting the effects of independent variables in terms of the effects on the probability of being a case versus being a control in one's sample typically has no substantive pertinence.

6.1.2 (Advanced) Interpretation using y^*

Binary logit and probit models are rarely interpreted in terms of the latent variable y^* . Accordingly, this section is primarily useful to provide a deeper understanding of identification and why logit coefficients are generally larger than probit coefficients.

As discussed in section 5.1.1, the logit and probit models can be derived from regression of a latent variable y^* :

$$y^* = \mathbf{x}\beta + \varepsilon$$

where ε is a random error. For the probit model, we assume ε is normal with $\text{Var}(\varepsilon) = 1$. For logit, we assume ε is distributed logistically with $\text{Var}(\varepsilon) = \pi^2/3$. As with the linear regression model, the marginal change in y^* with respect to x_k is

$$\frac{\partial y^*}{\partial x_k} = \beta_k$$

However, because y^* is latent, its true metric is unknown and depends on the identification assumption we make about the variance of the errors.

As we saw in section 5.2.2, the coefficients produced by **logit** and **probit** cannot be directly compared with one another. The logit coefficients will typically be about 1.7 times larger than the probit coefficients, simply as a result of the arbitrary assumption about the variance of the error. Consequently, the marginal change in y^* cannot be interpreted without standardizing by the estimated standard deviation of y^* , which is computed as

$$\hat{\sigma}_{y^*}^2 = \hat{\beta}' \widehat{\text{Var}}(\mathbf{x}) \hat{\beta} + \text{Var}(\varepsilon)$$

where $\widehat{\text{Var}}(\mathbf{x})$ is the covariance matrix for the observed x 's, $\hat{\beta}$ contains maximum likelihood estimates, and $\text{Var}(\varepsilon) = 1$ for probit and $\pi^2/3$ for logit. Then the y^* -standardized coefficient for x_k is

$$\beta_k^{Sy^*} = \frac{\beta_k}{\sigma_{y^*}}$$

which can be interpreted as follows:

For a unit increase in x_k , y^* is expected to increase by $\beta_k^{Sy^*}$ standard deviations, holding all other variables constant.

The fully standardized coefficient is

$$\beta_k^S = \frac{\sigma_k \beta_k}{\sigma_{y^*}}$$

which can be interpreted as follows:

For each standard deviation increase in x_k , y^* is expected to increase by β_k^S standard deviations, holding all other variables constant.

These coefficients are computed by `listcoef` with the `std` option:

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. listcoef, std
logit (N=753): Unstandardized and standardized estimates
Observed SD: 0.4956
Latent SD: 2.0474
Odds of: in LF vs not in LF
```

	b	z	P> z	bStdX	bStdY	bStdXY	SDofX
k5	-1.3916	-7.250	0.000	-0.729	-0.680	-0.356	0.524
k618	-0.0657	-0.961	0.336	-0.087	-0.032	-0.042	1.320
agecat							
40-49	-0.6268	-3.003	0.003	-0.305	-0.306	-0.149	0.487
50+	-1.2791	-4.924	0.000	-0.529	-0.625	-0.259	0.414
wc							
college	0.7977	3.481	0.001	0.359	0.390	0.175	0.450
hc							
college	0.1359	0.661	0.508	0.066	0.066	0.032	0.488
lwg	0.6099	4.045	0.000	0.358	0.298	0.175	0.588
inc	-0.0351	-4.238	0.000	-0.408	-0.017	-0.199	11.635
constant	1.0140	3.545	0.000

The y^* -standardized coefficients are in the column labeled `bStdY`, and the fully standardized coefficients are in the column `bStdXY`. We could interpret these coefficients as follows:

For each additional young child, the propensity of a women to join the labor force decreases by 0.68 standard deviations, holding all other variables constant.

For every standard deviation increase in family income, a woman's propensity to join the labor force is expected to decrease by 0.199 standard deviations, holding all other variables constant.

Next, we compute the y^* -standardized coefficients for probit:


```
. probit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. listcoef, std
probit (N=753): Unstandardized and standardized estimates
Observed SD: 0.4956
Latent SD: 1.1530
```

	b	z	P> z	bStdX	bStdY	bStdXY	SDofX
k5	-0.8396	-7.503	0.000	-0.440	-0.728	-0.382	0.524
k618	-0.0412	-1.011	0.312	-0.054	-0.036	-0.047	1.320
agecat							
40-49	-0.3816	-3.058	0.002	-0.186	-0.331	-0.161	0.487
50+	-0.7801	-5.000	0.000	-0.323	-0.677	-0.280	0.414
wc							
college	0.4821	3.555	0.000	0.217	0.418	0.188	0.450
hc							
college	0.0738	0.596	0.551	0.036	0.064	0.031	0.488
lwg	0.3710	4.211	0.000	0.218	0.322	0.189	0.588
inc	-0.0211	-4.368	0.000	-0.245	-0.018	-0.212	11.635
constant	0.6222	3.692	0.000

Although the estimates of β in column **b** are uniformly smaller than those from logit, the y^* -standardized and fully standardized coefficients in columns **bStdY** and **bStdXY** are very similar, which demonstrates that the differences in the magnitude of coefficients in logit and probit are due to differences in scale.

An issue related to y^* -standardized coefficients arises when researchers compare coefficients across models. In the linear regression model, mediating variables are often added to a model, and the change in coefficients is interpreted as indicating how much of the effect of an independent variable on the dependent variable is due to the indirect effect of the mediating variable (see, for example, Breen, Karlson, and Holm [2013, 166–167]). For example, if the coefficient estimating the effect of childhood socioeconomic status (SES) on adult earnings is reduced when educational attainment is added to the model, one might say that half the effect of SES on earnings is explained by educational attainment.

This interpretation of a change in unstandardized logit or probit coefficients is problematic (Winship and Mare 1984). In linear regression, when independent variables are added to a model, $\widehat{\text{Var}}(\mathbf{x}\hat{\beta})$ increases and $\widehat{\text{Var}}(\varepsilon)$ decreases accordingly, because the observed variance of y must remain the same. In the logit and probit models, when independent variables are added to a model, $\widehat{\text{Var}}(\mathbf{x}\hat{\beta})$ increases but $\widehat{\text{Var}}(\varepsilon)$ does not change because its value is assumed. Consequently, $\widehat{\text{Var}}(y^*)$ must increase. For the BRM, the indirect effects interpretation across models with different independent variables no longer holds because as the model specification changes, the scale of the latent dependent variable changes.

One can think about y^* -standardized coefficients as the coefficients we would observe if $\text{Var}(y^*)$ was rescaled to be fixed to 1. If so, then in terms of y^* -standardized coefficients, adding new independent variables to the model does not increase this rescaled $\text{Var}(y^*)$. For indirect effects, interpreting changes in y^* -standardized logit and probit coefficients seems clearly preferable to doing so for unstandardized coefficients. Alternatives to y^* -standardization have also been proposed (Karlson, Holm, and Breen 2012; Breen and Karlson 2013).

6.2 Marginal effects: Changes in probabilities

A marginal effect measures the change in the probability of an outcome for a change in x_k , holding all other independent variables constant at specific values. The critical idea is that one variable is changing while the other variables are not. There are two varieties of marginal effects. A marginal change computes the effect of an instantaneous or infinitely small change in x_k . A discrete change computes the effect of a discrete or finite change in x_k . (See section 4.5 for an introductory discussion of marginal effects.)

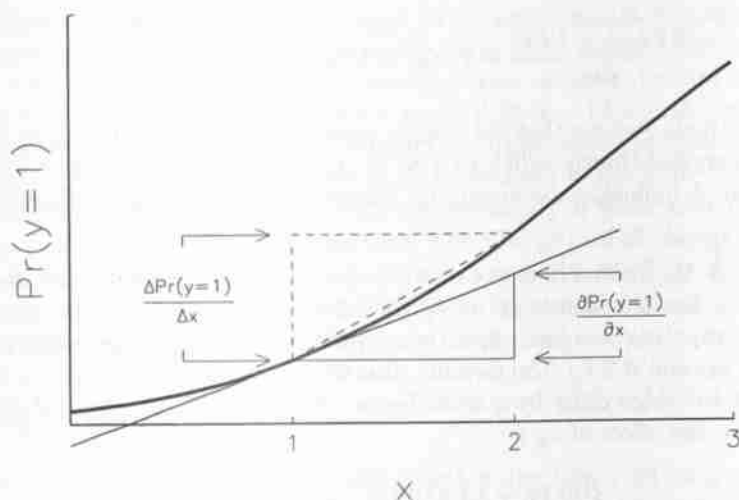


Figure 6.1. Marginal change and discrete change in the BRM

A marginal change, shown by the tangent to the probability curve at $x = 1$ in figure 6.1, is the rate of change in the probability for a infinitely small change in x_k , holding other variables at specific values:

$$\frac{\partial \Pr(y = 1 \mid \mathbf{x} = \mathbf{x}^*)}{\partial x_k}$$

Because the effect is computed with a partial derivative, some authors refer to this as the partial change or partial effect. In this formula, \mathbf{x}^* contains specific values of the independent variables. For example, \mathbf{x}^* could equal \mathbf{x}_i with the observed values for the i th observation, it could equal the means $\bar{\mathbf{x}}$ of all variables, or it could equal any other values. When the meaning is clear, we will refer to \mathbf{x} without specifying \mathbf{x}^* . The important thing is that the value of the marginal effect depends on the specific values of the x_k 's where the change is computed.

In the BRM, the marginal change has the simple formula

$$\frac{\partial \Pr(y_i = 1 \mid \mathbf{x})}{\partial x_k} = f(\mathbf{x}\beta) \beta_k$$

where f is the normal probability distribution function (PDF) for probit and the logistic PDF for logit. In logit models, the marginal change has a particularly convenient form:

$$\frac{\partial \Pr(y_i = 1 \mid \mathbf{x})}{\partial x_k} = \Pr(y_i = 1 \mid \mathbf{x}) [1 - \Pr(y_i = 1 \mid \mathbf{x})] \beta_k$$

From this formula, we see that the change must be greatest when $\Pr(y = 1 \mid \mathbf{x}) = 0.5$, where the marginal change is $(0.5)(0.5)\beta_k = \beta_k/4$. Accordingly, dividing a binary logit coefficient by 4 indicates the maximum marginal change in the probability (Cramer 1991, 8).

As long as the model does not include power or interaction terms, the marginal change for x_k has the same sign as β_k for all values of \mathbf{x} because the PDF is always positive. (Computing marginal effects when powers and interactions are in the model is discussed in section 6.2.1.) The formula also shows that marginal changes for different independent variables differ by a scale factor. For example, the ratio of the marginal effect of x_j to the effect of x_k is

$$\frac{\partial \Pr(y_i = 1 \mid \mathbf{x}) / \partial x_j}{\partial \Pr(y_i = 1 \mid \mathbf{x}) / \partial x_k} = \frac{f(\mathbf{x}\beta) \beta_j}{f(\mathbf{x}\beta) \beta_k} = \frac{\beta_j}{\beta_k}$$

for all values of \mathbf{x} . Consequently, while β_k does not tell you the magnitude of x_k 's effect, it can tell you how much larger or smaller it is than the effects of other variables.

A discrete change, sometimes called a first difference, is the actual change in the predicted probability for a given change in x_k , holding other variables at specific values. For example, the discrete change for an increase in age from 30 to 40 is the change in the probability of being in the labor force as age increases from 30 to 40, holding other

variables at specified values. Defining x_k^{start} as the starting value of x_k and x_k^{end} as the ending value, the discrete change equals

$$\frac{\Delta \Pr(y = 1 | \mathbf{x})}{\Delta x_k (x_k^{\text{start}} \rightarrow x_k^{\text{end}})} = \Pr(y = 1 | \mathbf{x}, x_k = x_k^{\text{end}}) - \Pr(y = 1 | \mathbf{x}, x_k = x_k^{\text{start}})$$

For binary variables, such as having attended college, the obvious choice is a change from 0 to 1:

$$\frac{\Delta \Pr(y_i = 1 | \mathbf{x})}{\Delta x_k (0 \rightarrow 1)} = \Pr(y_i = 1 | x_k^{\text{end}} = 1, \mathbf{x}) - \Pr(y_i = 1 | x_k^{\text{start}} = 0, \mathbf{x})$$

We often are interested in the discrete changes when a variable increases by some amount δ from its observed value. Defining $\Pr(y = 1 | \mathbf{x}, x_k)$ as the probability at \mathbf{x} , noting in particular the value of x_k , the discrete change for a change of δ in x_k equals

$$\frac{\Delta \Pr(y = 1 | \mathbf{x})}{\Delta x_k (x_k \rightarrow x_k + \delta)} = \Pr(y = 1 | \mathbf{x}, x_k + \delta) - \Pr(y = 1 | \mathbf{x}, x_k)$$

We might want to examine a discrete change of one unit, a standard deviation, 15 points for IQ, 4 years for education, the range of income, or 10 years for age.

The discrete change tells you how much the probability actually changes for a given change in a variable. To the degree that the probability curve is linear in the region where the change occurs, the marginal change for x_k approximates the discrete change for a unit increase in x_k . The more nonlinear the curve in the region where x_k increases, the greater the difference between the marginal change and the discrete change. Because in general $\partial \Pr(y = 1 | \mathbf{x}) / \partial x_k$ does not equal $\Delta \Pr(y = 1 | \mathbf{x}) / \Delta x_k$, we prefer the discrete change that indicates the actual amount of change in the probability for a specific change in x_k . For example, we find it more meaningful to say that “for a standard deviation increase in income, about \$11,000, the probability of labor force participation decreases on average by 0.09” than to say that “the average rate of change in the probability of labor force participation with respect to income is -0.007 ”. However, some fields, such as economics, have a strong preference for marginal change over discrete change for continuous independent variables.

6.2.1 Linked variables

Fundamental to the concept of a marginal effect is the idea that only one variable changes while holding all other variables at specified values. An exception must be made for variables that are linked mathematically. For example, if x_{age} is age and $x_{\text{agesq}} = x_{\text{age}} \times x_{\text{age}}$, you cannot change x_{age} while holding x_{agesq} constant. The change in x_{age} must be matched by a corresponding change in x_{agesq} . This is easy to illustrate with a discrete change in age from 20 to 30:

$$\begin{aligned} \frac{\Delta \Pr(y = 1 | \mathbf{x})}{\Delta \text{age} (20 \rightarrow 30)} &= \Pr(y = 1 | \mathbf{x}, x_{\text{age}} = 30, x_{\text{agesq}} = 30^2) \\ &\quad - \Pr(y = 1 | \mathbf{x}, x_{\text{age}} = 20, x_{\text{agesq}} = 20^2) \end{aligned}$$

Linked variables must also be considered for the variables being held constant. For example, if we are computing the marginal effect of x_k while holding age at its mean, we need to hold x_{age} at $\text{mean}(x_{\text{age}})$ and x_{agesq} at $[\text{mean}(x_{\text{age}}) \times \text{mean}(x_{\text{age}})]$ not at $\text{mean}(x_{\text{agesq}})$. Similarly, if your model includes x_{female} , x_{age} , and the interaction $x_{\text{female} \times \text{age}} = x_{\text{female}} \times x_{\text{age}}$, you cannot change x_{age} while holding $x_{\text{female} \times \text{age}}$ constant.

Categorical regressors that enter a model as a set of indicators are also linked. Suppose that education has three categories: no high school degree, high school diploma as the highest degree, and college diploma as the highest degree. Let $x_{\text{hs}} = 1$ if high school is the highest degree and equal 0 otherwise; and let $x_{\text{college}} = 1$ if college is the highest degree and equal 0 otherwise. If $x_{\text{hs}} = 1$, then $x_{\text{college}} = 0$. You cannot increase x_{college} from 0 to 1 while holding x_{hs} at 1. Computing the effect of having college as the highest degree ($x_{\text{hs}} = 0, x_{\text{college}} = 1$) compared with high school as the highest degree ($x_{\text{hs}} = 1, x_{\text{college}} = 0$) involves changing two variables:

$$\frac{\Delta \Pr(y = 1 \mid \mathbf{x})}{\Delta x_{\text{hs}}(0 \rightarrow 1) \ \& \ x_{\text{college}}(1 \rightarrow 0)} = \Pr(y = 1 \mid \mathbf{x}, x_{\text{hs}} = 0, x_{\text{college}} = 1) - \Pr(y = 1 \mid \mathbf{x}, x_{\text{hs}} = 1, x_{\text{college}} = 0)$$

When discussing marginal effects with linked variables, we will say “holding other variables constant” with the implicit understanding that appropriate adjustments for linked variables are being made. A major benefit of using factor-variable notation when specifying a regression model is that `margins`, `mchange`, `mtable`, and `mgen` keep track of which variables are linked, and compute predictions and marginal effects correctly.

6.2.2 Summary measures of change

The marginal effect of a variable depends on the specific values of all independent variables. Because the effect of x_k differs for each observation (unless, of course, multiple observations have identical values), there is a distribution of marginal effects in the sample. For interpretation, we seek a simple, informative summary of this distribution of effects. There are three basic approaches:

Marginal effect at the mean (MEM). Compute the marginal effect of x_k with all variables held at their means.

Marginal effect at representative values (MER). Compute the marginal effect of x_k with variables held at specific values that are selected for being especially instructive for the substantive questions being considered. The MEM is a special case of the MER.

Average marginal effect (AME). Compute the marginal effect of x_k for each observation at its observed values \mathbf{x}_i , and then compute the average of these effects.

We consider each measure before discussing how to decide which measure is appropriate for your application.

MEMs and MERs

The MEM is computed with all variables held at their means. For a marginal change, this is

$$\frac{\partial \Pr(y = 1 \mid \bar{\mathbf{x}}, x_k = \bar{x}_k)}{\partial x_k}$$

which can be interpreted as follows:

For someone who is average on all characteristics, the marginal change of x_k is ...

The discrete change equals

$$\frac{\Delta \Pr(y = 1 \mid \bar{\mathbf{x}}, x_k = \bar{x}_k)}{\Delta x_k}$$

and can be interpreted as follows:

For someone who is average on all characteristics, increasing x_k by δ changes the probability by ...

The MER would replace "who is average" with a description of the values of the covariates.

AMEs

The AME is the mean of the marginal effect computed at the observed values for all observations in the estimation sample. For a marginal change, this is

$$\text{mean} \frac{\partial \Pr(y_i = 1 \mid \mathbf{x}_i)}{\partial x_k} = \frac{1}{N} \sum_{i=1}^N \frac{\partial \Pr(y_i = 1 \mid \mathbf{x} = \mathbf{x}_i)}{\partial x_k}$$

which can be interpreted as follows:

The average marginal effect of x_k is ...

The average discrete change equals

$$\text{mean} \frac{\Delta \Pr(y_i = 1 \mid \mathbf{x}_i)}{\Delta x_k} = \frac{1}{N} \sum_{i=1}^N \frac{\Delta \Pr(y_i = 1 \mid \mathbf{x} = \mathbf{x}_i)}{\Delta x_k}$$

which is interpreted as follows:

On average, increasing x_k by δ increases the probability by ...

For factor variables or changes from one fixed value to another (for example, the minimum to the maximum), we say

On average, increasing x_k from *start-value* to *end-value* increases the probability by ...

Standard errors of marginal effects

For each of these measures of change, standard errors can be computed using the delta method (Agresti 2013, 72–75; Wooldridge 2010, 576–577; Xu and Long 2005; [R] **margins**). The standard errors allow you to test whether a marginal effect is 0, to add a confidence interval to the estimated effect, and to test such things as whether marginal effects are equal at different values of the independent variables.

6.2.3 Should you use the AME, the MEM, or the MER?

No summary measure of effects is ideal for all situations, but how do you decide which measure to use? Since the 1980s, the literature has provided weak recommendations for AMEs, but our reading suggests that AMEs were rarely used. In his classic book on limited and qualitative dependent variables, Maddala (1983, 24) expressed reservations about MEMs because marginal effects vary by the level of the variables. He suggested that “we need to calculate [marginal effects] at different levels of the explanatory variables to get an idea of the range of variation of the resulting changes in probabilities”. Essentially, he is recommending computing multiple MERs at substantively informative locations in the data. Because the effect of a variable differs at different places in the data, multiple MERs provide insights into the magnitude and variation of the effects. Long (1997, 74) wrote that “since \bar{x} might not correspond to any observed values in the population, averaging over observations might be preferred”. Cameron and Trivedi (2005, 467) suggest that “it is best to use AME over MEM”. Hanmer and Kalkan (2013) argue that “the observed-value approach [(AME)] is preferable to the more common average-case approach [(MEM)] on theoretical grounds”.

The popularity of the MEM is probably because of ease of computation. Computing an AME in principle involves N times more computation than the corresponding MEM. With the rapid growth in computing power, this is a trivial issue compared with having readily available software that easily computes the AME. For example, with our **prchange** command in **SPost9**, computing MEMs was trivially easy. Although you could compute the AME with **prchange**, you needed to write your own program to collect and summarize the computations for each observation. Few people, ourselves included, bothered to do that. With Stata’s **margins** and our **mchange**, it is as easy to compute AMEs as MEMs.¹ These computational advances do not, however, imply that the AME

1. Surprisingly, **margins** actually computes the AME faster than the MEM because it always computes the AME before computing an MEM or MER. To compute the MEM for a dataset with 50,000 observations, **margins** will compute 50,000 marginal changes you do not need before it computes the one you do.

is always the best way to assess the effect of a variable. There are several issues to consider when deciding which measure of change to use.

Does the marginal effect computed at the mean of all variables provide useful information about the overall effect of that variable? This is relevant not only in deciding what to do in current analyses, but when evaluating past research that used the MEM. A common criticism of the MEM is that typically there is no actual case in the dataset for which all variables equal the mean. Most obviously, with binary independent variables, the mean does not correspond to a possible value of an observation. For example, a variable like *pregnant* is measured as 0 and 1 without it being possible to observe someone with a value equal to a sample mean of intermediate value. This issue alone leads some to disfavor the MEM (Hanmer and Kalkan 2013). We are not ourselves as concerned about this point because holding a binary variable at its mean is, roughly speaking, taking a weighted mean of effects for each group. If the groups are a focus of the analysis, you can compute MERs for each group by using group-specific means. Alternatively, effects can be computed at the modal values of the binary variables, but this ignores everyone who is in a less well-represented group.

Sometimes, it is argued that the MEM is a reasonable approximation to the AME. Although Greene and Hensher (2010, 36) correctly observed that the AME and MEM are often similar, they incorrectly suggest that this is especially true in large samples. Although the two measures will often be similar, they can differ in substantively meaningful ways, and whether this is the case has little to do with whether a sample is bigger or smaller.

Bartus (2005) and Verlinda (2006) explain more precisely when MEM and AME differ and which is larger. For the binary logit and probit models, the difference between the AME and MEM for x_k depends on three things: the probability that $y = 1$ when all x_k 's are held to their means, the variance of $\mathbf{x}\beta$, and the size of β_k (Bartus 2005; Hanmer and Kalkan 2013, S1). The sign of the difference between the AME and MEM depends on $\Pr(y = 1 \mid \bar{\mathbf{x}})$, with the AME being larger at lower and higher probabilities. In the middle, the MEM is larger, with the largest difference occurring when $\Pr(y = 1 \mid \bar{\mathbf{x}}) = 0.5$. The AME and MEM will be equal when the probability is about 0.21 and 0.79 for the binary logit model, and about 0.15 and 0.85 for the binary probit model.

The AME, MEM, and MER are each summary measures, and no single summary of effects is ideal for all situations. Broadly speaking, we believe that the AME is the best summary of the effect of a variable. Because it averages the effects across all cases in the sample, it can be interpreted as the average size of the effect in the sample. The MEM is computed at values of the independent variables that might not be representative of anyone in the sample.

However, both AME and MEM are limited because they are based on averages. If the average value of each regressor is a substantively interesting location in the data, the MEM is useful because it tells you the magnitude of the effect for someone with those or similar characteristics. If the average of the independent variables is not an interesting location, it is not useful. If you are interested in the average effect in the sample, the

AME is appropriate. However, it is possible that nobody in the sample has a marginal effect that is close to the AME. If, for example, you are interested in the effect of a treatment for young minority women, knowing the effect for someone who is average is not helpful. Similarly, the average effect of the treatment for the entire sample does not tell you the effect for the group of young minority women you are interested in. In general, the AME is not necessarily more informative than a set of MERs computed at substantively interesting places. Or, as shown in section 6.2.5, you can examine the distribution of effects for all observations.

No single number is a substitute for understanding how predictions vary over the range of one's data and for conveying the fact of that variation when it is substantively meaningful. Thus the best measure is the one that addresses the goals of your research. Although examining AMEs of your independent variables is an important step in data analysis, this should be followed by a more detailed analysis of predictions in tables or graphs.

6.2.4 Examples of marginal effects

In this section, we use our model of labor force participation to illustrate the computation and interpretation of marginal effects with `mchange`. The `mchange` command makes it simple to compute marginal effects for different amounts of changes, either averaging effects over the sample or computing them at fixed values. `mchange` uses `margins` to compute the effects, which are then collected into a compact table. For example, running `mchange` after fitting our baseline model creates a 30-line table that summarize 500 lines of output from a dozen `margins` commands. If you want to learn more about `margins`, you can add the option `details` to `mchange` to see how to use `margins` output. Information on using `margins` to compute marginal effects is given in section 6.2.6.

We begin by fitting our model and storing the estimates so that they can be restored later:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
Logistic regression
Number of obs = 753
LR chi2(8) = 124.30
Prob > chi2 = 0.0000
Pseudo R2 = 0.1207
Log likelihood = -452.72367
```

lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
k5	-1.391567	.1919279	-7.25	0.000	-1.767739	-1.015395
k618	-.0656678	.068314	-0.96	0.336	-.1995607	.0682251
agecat						
40-49	-.6267601	.208723	-3.00	0.003	-1.03585	-.2176705
50+	-1.279078	.2597827	-4.92	0.000	-1.788242	-.7699128
wc						
college	.7977136	.2291814	3.48	0.001	.3485263	1.246901
hc						
college	.1358895	.2054464	0.66	0.508	-.266778	.5385569
lwg	.6099096	.1507975	4.04	0.000	.314352	.9054672
inc	-.0350542	.0082718	-4.24	0.000	-.0512666	-.0188418
_cons	1.013999	.2860488	3.54	0.000	.4533539	1.574645

```
. estimate store base
```

The descriptive statistics for the estimation sample are

```
. estat summarize, labels
Estimation sample logit
Number of obs = 753
```

Variable	Mean	Std. Dev.	Min	Max	Label
lfp	.5683931	.4956295	0	1	In paid labor force?
k5	.2377158	.523959	0	3	# kids < 6
k618	1.353254	1.319874	0	8	# kids 6-18
agecat					Wife's age group
40-49	.3851262	.4869486	0	1	
50+	.2191235	.4139274	0	1	
wc					Wife attended college?
college	.2815405	.4500494	0	1	
hc					Husband attended college?
college	.3917663	.4884694	0	1	
lwg	1.097115	.5875564	-2.05412	3.21888	Log of wife's estimated wages
inc	20.12897	11.6348	-.029	96	Family income excluding wife's

We will next show how to compute and interpret AMEs for continuous and factor variables, before examining the corresponding MEMs. Marginal effects in models with powers and interactions are then considered. Finally, we show how to compute the distribution of effects for observations in the estimation sample.

AMEs for continuous variables

For continuous independent variables, `mchange` computes the average marginal change and average discrete change of 1 and a standard deviation. To assess the effects of income and wages, type²

```
. mchange inc lwg
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

		Change	p-value
inc			
	+1	-0.007	0.000
	+SD	-0.086	0.000
	Marginal	-0.007	0.000
lwg			
	+1	0.120	0.000
	+SD	0.072	0.000
	Marginal	0.127	0.000
Average predictions			
		not in LF	in LF
Pr(y base)		0.432	0.568

The average predictions, listed below the table of changes, show that in the sample the average predicted probability of being in the labor force is 0.432. This is the same value you would obtain by first running `predict` and then computing the mean of the predictions. In later examples, we often suppress this result by adding the `brief` option. Summarizing the AMEs for a standard deviation change, we can say

Holding other variables at their observed values, increasing income by one standard deviation, roughly \$12,000, decreases the probability of labor force participation on average by 0.09. An increase of one standard deviation in the log of anticipated wages, about 0.6, increases the probability by 0.07. Both effects are significant at the 0.001 level.

There are two points of interest here. First, the marginal and unit discrete changes are similar, which reflects that the probability curve is nearly linear for a change of 1

2. In Stata 12 and earlier, standard errors are not available for average discrete changes of a fixed amount of change (for example, 1 or a standard deviation) from the observed value. Based on our experience with Stata 13, which can compute standard errors for these effects, we believe that the significance level for the marginal change is a good approximation, especially when the values of the marginal change and discrete change are similar.

in either variable. Second, for `lwg`, the marginal change and unit discrete change are potentially misleading because a unit change in `lwg` corresponds to a change of nearly two standard deviations.

The `delta(#)` option allows us to compute the effect of a change of any amount `#` to replace the default change of one standard deviation. For example, to compute the effect of a \$5,000 change in income, we use `delta(5)`:

```
. mchange inc, delta(5) brief
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

		Change	p-value
inc			
	+1	-0.007	0.000
	+delta	-0.037	0.000
	Marginal	-0.007	0.000

This can be interpreted as follows:

On average, an increase of \$5,000 in income decreases the probability of labor force participation by 0.04 ($p < 0.001$).

Note that our reporting of results has become shorter by no longer making explicit that other variables are kept at their observed values.

For the number of young children, `k5`, the most reasonable effect is for an increase of one child, so we specify `amount(one)`:

```
. mchange k5, amount(one) brief
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

		Change	p-value
k5			
	+1	-0.281	0.000

We conclude the following:

On average, holding other variables at their observed values, increasing the number of young children in a family by one is expected to decrease the probability of labor force participation by 0.28 ($p < 0.001$, two-tailed).

A quick way to assess the maximum potential impact of continuous variables (that is, nonfactor variables) is to compute the AME over the range. Changes over the range tell us how much you would expect the outcome probability to change in the unlikely event of massive changes in a variable, holding other variables constant. For example, what would happen if a person increased her family income from \$0 to \$104,000, without other variables changing? Although this type of massive change is not likely to occur in

the real world, it provides a bound for the largest effect you could find. If the change is small and nonsignificant, the lack of effect might be substantively interesting, but you are unlikely to learn anything more about the variable by analyzing it further.

Consider the effects of changing from 0 to the maximum number of young or older children in the sample:

```
. mchange k5 k618, amount(range) brief
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

		Change	p-value
k5			
	Range	-0.599	0.000
k618			
	Range	-0.110	0.336

Using information on the range from the summary statistics shown above, we see that changing from 0 to 3 young children has a very large and significant AME of -0.60 . Changing from 0 to 8 older children, on the other hand, has a nonsignificant effect. Accordingly, k618 will not be considered further in our examples of interpretation.

Because a variable's range can be influenced by even a single extreme observation (for example, one person with an unusually high income), we suggest using a trimmed range for some variables. For example, `mchange inc lwg, amount(range) trim(5)` computes the AME for a change in income and anticipated wages from the 5th percentile to the 95th percentile:

```
. mchange inc lwg, amount(range) trim(5) brief
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

		Change	p-value
inc			
	5% to 95%	-0.249	0.000
lwg			
	5% to 95%	0.239	0.000

We can interpret the discrete change for `inc` as follows:

On average the probability of labor force participation will decrease by 0.25 if respondents changed from the 5th percentile of income to the 95th percentile ($p < 0.001$, two-tailed).

We could interpret `lwg` similarly.

To fully understand the meaning of a discrete change over the range, we need to know the range over which a variable varies. We obtain this information by using `centile varlist, centile(0 5 95 100)` to request the 5th and 95th percentiles along with the minimum and maximum. Here we request information for `inc` and `lwg`:

```
. centile inc lwg, centile(0 5 95 100)
```

Variable	Obs	Percentile	Centile	— Binom. Interp. — [95% Conf. Interval]	
inc	753	0	-.0290001	-.0290001	-.0290001*
		5	7.0428	6.334469	7.789306
		95	41.19	37.50825	45.02423
		100	96	96	96*
lwg	753	0	-2.054124	-2.054124	-2.054124*
		5	.2065675	.1047049	.3321936
		95	2.084091	1.953158	2.153738
		100	3.218876	3.218876	3.218876*

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

Comparing the 95th with the 100th percentile shows that for both variables, the trimmed range excludes extreme observations.

We can obtain more information about the discrete changes by using the option `statistics()` to request the starting and ending probabilities. To show this, we compute the changes for income and log of wages:

```
. mchange inc lwg, amount(range) trim(5)
> statistics(change from to pvalue) brief
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

	Change	From	To	p-value
inc				
5% to 95%	-0.249	0.660	0.411	0.000
lwg				
5% to 95%	0.239	0.454	0.693	0.000

Using the new information, we can elaborate our interpretation of the effect of income:

Changing family income from its 5th percentile of \$7,000 to the 95th percentile of \$41,000 on average decreases the probability of a woman being in the labor force from 0.66 to 0.41, a decrease of 0.25 ($p < 0.001$, two-tailed).

The effect for `lwg` could be interpreted similarly.

AMEs for factor variables

For binary independent variables, the only reasonable change is from 0 to 1, which is the default when factor-variable notation is used. Because `hc` and `wc` were included in the logit specification as `i.hc` and `i.wc`, `mchange` automatically computes a discrete change from 0 to 1:


```
. mchange hc wc, stat(change from to pvalue) brief
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

	Change	From	To	p-value
hc				
college vs no	0.028	0.558	0.586	0.508
wc				
college vs no	0.162	0.525	0.688	0.000

We interpret these results as follows:

On average, having attended college increases a woman's probability of labor force participation from 0.53 to 0.69, a change of 0.16 ($p < 0.001$), while the effect of the husband having attended college is not significant.

The `mchange` command also works with factor variables that have more than two categories. Because `agecat` has three categories and was entered into our model as `i.agecat`, `mchange` computes effects of changes between all categories, referred to as contrasts:

```
. mchange agecat, stat(change from to pvalue) brief
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

	Change	From	To	p-value
agecat				
40-49 vs 30-39	-0.124	0.676	0.552	0.002
50+ vs 30-39	-0.262	0.676	0.414	0.000
50+ vs 40-49	-0.138	0.552	0.414	0.002

Each contrast can be interpreted exactly as the interpretation of a binary independent variable. The discrete change labeled 40-49 vs 30-39 is the effect of being in the age group 40-49 compared with being in the age group 30-39, and so on for other comparisons.

On average, being 40 to 49 compared with being 30 to 39 decreases the probability of labor force participation by 0.12 ($p < 0.01$). Being 50 or older compared with being 30 to 39 decreases the probability by 0.26 ($p < 0.001$). Being 50 or older compared with being 40 to 49 decreases the probability by 0.14 ($p < 0.01$).

Notice that knowing two of the contrasts implies the third: $0.124 + 0.138 = 0.262$.

Summary table of AMEs

Computing AMEs is often the next step after examining predictions with `predict`. AMEs quickly provide a general sense of the effects of each variable, much like regression

coefficients in linear regression. After fitting your model, `mchange` with the default options provides a quick summary:

```
. mchange
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

		Change	p-value
k5			
	+1	-0.281	0.000
	+SD	-0.153	0.000
	Marginal	-0.289	0.000
k618			
	+1	-0.014	0.337
	+SD	-0.018	0.337
	Marginal	-0.014	0.335
agecat			
	40-49 vs 30-39	-0.124	0.002
	50+ vs 30-39	-0.262	0.000
	50+ vs 40-49	-0.138	0.002
wc			
	college vs no	0.162	0.000
hc			
	college vs no	0.028	0.508
lwg			
	+1	0.120	0.000
	+SD	0.072	0.000
	Marginal	0.127	0.000
inc			
	+1	-0.007	0.000
	+SD	-0.086	0.000
	Marginal	-0.007	0.000

Average predictions

	not in LF	in LF
Pr(y base)	0.432	0.568

If you are not using marginal changes, you can specify the amount of change and decimal places:

```
. mchange, amount(one sd) decimals(2) brief
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

		Change	p-value
k5			
	+1	-0.28	0.00
	+SD	-0.15	0.00
k618			
	+1	-0.01	0.34
	+SD	-0.02	0.34
agecat			
40-49 vs 30-39		-0.12	0.00
50+ vs 30-39		-0.26	0.00
50+ vs 40-49		-0.14	0.00
wc			
college vs no		0.16	0.00
hc			
college vs no		0.03	0.51
lwg			
	+1	0.12	0.00
	+SD	0.07	0.00
inc			
	+1	-0.01	0.00
	+SD	-0.09	0.00

If you prefer marginal changes instead of discrete changes, you could use `mchange`, `amount(marginal)` instead. With marginal changes, you will probably need at least three decimal places. If you prefer discrete changes that are centered, use the `centered` option.

We find AMEs so much more useful than the estimated β 's or odds ratios that we wish the standard output from logit or probit would present AMEs along with estimated coefficients, perhaps as an option (like the `or` option to `logit` presents odds ratios instead of untransformed coefficients).

Marginal effects for subgroups

We might be interested in comparing marginal effects for subgroups in our sample. For example, we might be interested in the effect of an additional child on labor force participation for women who have attended college. In this case, all the same arguments that we made earlier for why we broadly prefer AMEs to MEMs apply, only now we are applying these arguments to a subgroup of our sample instead of to the whole sample. As a result, we want the average change only over members of this subgroup. We can obtain this by using an `if` condition:


```
. mchange k5 if wc==1, amount(one)
logit: Changes in Pr(y) | Number of obs = 212
Expression: Pr(lfp), predict(pr)
```

		Change	p-value
k5			
	+1	-0.276	0.000

Average predictions

	not in LF	in LF
Pr(y base)	0.321	0.679

```
1: Sample selection: if wc==1 & e(sample)==1
```

We can interpret this as follows:

For women who have attended college, an additional child decreases the probability of being in the labor force by an average of 0.28.

In section 6.5, after we learn more about `mtable`, we show how to compute AMEs for different groups and how to test whether the marginal effects are equal across groups.

MEMs and MERs

Although we tend to prefer AMEs as a summary measure of change, marginal effects are often computed at the mean or at other values. With the `atmeans` option, `mchange` computes the MEMs. We will use `atmeans` extensively in section 6.3, when we discuss generating predictions based upon the hypothetical observations we call ideal types. A hypothetical observation implies specific values for all values of the independent variables, each of which we can either specify directly with `at()` or specify by using `atmeans`. Talking about a hypothetical observation can be contrasted with the example of subgroup analysis immediately above, in which we were making statements about a group of observations and wanted to compute the average effect over that group.

Here we compute marginal effects by using `atmeans`, adding the `statistics(ci)` option to request the confidence intervals rather than the p -values.


```
. mchange, statistics(ci) atmeans
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

		Change	LL	UL
k5				
	+1	-0.324	-0.396	-0.252
	+SD	-0.180	-0.227	-0.133
	Marginal	-0.339	-0.431	-0.247
k618				
	+1	-0.016	-0.049	0.017
	+SD	-0.021	-0.065	0.022
	Marginal	-0.016	-0.049	0.017
agecat				
	40-49 vs 30-39	-0.146	-0.238	-0.053
	50+ vs 30-39	-0.307	-0.422	-0.191
	50+ vs 40-49	-0.161	-0.265	-0.058
wc				
	college vs no	0.186	0.088	0.284
hc				
	college vs no	0.033	-0.065	0.131
lwg				
	+1	0.138	0.078	0.198
	+SD	0.084	0.045	0.123
	Marginal	0.149	0.077	0.221
inc				
	+1	-0.009	-0.013	-0.005
	+SD	-0.101	-0.148	-0.054
	Marginal	-0.009	-0.013	-0.005

Predictions at base value

	not in LF	in LF
Pr(y base)	0.422	0.578

Base values of regressors

	k5	k618	2. agecat	3. agecat	1. wc	1. hc
at	.238	1.35	.385	.219	.282	.392
	lwg	inc				
at	1.1	20.1				

1: Estimates with margins option atmeans.

The values at which variables are being held constant are listed in the table **Base values of regressors**.

Here are examples of interpreting each type of effect:

Change of 1 at the mean. For a woman who is average on all characteristics, an additional young child decreases the probability of being in the labor force by 0.32 (95% CI: [0.25, 0.40]).

Change of standard deviation at the mean. A standard deviation increase in income, roughly \$11,000, decreases the probability of being in the labor force by 0.10 (95% CI: [0.05, 0.15]), holding other variables at their means.

Change from 0 to 1 at the mean. If a woman attended college, her probability of being in the labor force is 0.19 greater than a woman who did not attend college, holding other variables at their means (95% CI: [0.09, 0.28]).

Change of categorical variables at the mean. For an average woman, being 40 to 49 compared with being 30 to 39 decreases the probability of being in the labor force by 0.15 (95% CI: [0.05, 0.24]). Being 50 or older compared with being 30 to 39 decreases the probability by 0.31 (95% CI: [0.19, 0.42]). Being 50 or older compared with being 40 to 49 decreases the probability by 0.16 (95% CI: [0.06, 0.27]).

Marginal effects can be computed at other values by using the `at()` option. For example, to compute the marginal effect of `k5` for a hypothetical family in which the husband and wife both attended college, holding other variables at their means:

```
. mchange k5, at(wc=1 hc=1) amount(1) atmeans
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

	Change	p-value				
k5						
+1	-0.329	0.000				
Predictions at base value						
	not in LF	in LF				
Pr(y base)	0.275	0.725				
Base values of regressors						
	k5	k618	2. agecat	3. agecat	wc	hc
at	.238	1.35	.385	.219	1	1
	lwg	inc				
at	1.1	20.1				

1: Estimates with margins option `atmeans`.

Notice that `hc` and `wc` are listed as 1 in the table of base values. We conclude the following:

For an otherwise average family in which the husband and wife both attended college, an additional young child decreases the probability of being in the labor force by 0.33 ($p < 0.001$).

Or, we might want to estimate the effect of changing income from \$0 to \$5,000 for a family with two young children and in which neither parent went to college, holding

other variables at their means. We set the values of the variables by using `at(inc=0 k5=2 wc=0 hc=0) atmeans`, and we indicate that we want a change of five by using `delta(5)`.

```
. mchange inc, delta(5) amount(delta) at(inc=0 k5=2 wc=0 hc=0) atmeans
```

```
logit: Changes in Pr(y) | Number of obs = 753
```

```
Expression: Pr(lfp), predict(pr)
```

	Change	p-value				
inc						
+delta	-0.021	0.012				
Predictions at base value						
	not in LF	in LF				
Pr(y base)	0.847	0.153				
Base values of regressors						
	k5	k618	2. agecat	3. agecat	wc	hc
at	2	1.35	.385	.219	0	0
	lwg	inc				
at	1.1	0				

1: Estimates with margins option `atmeans`.

2: Delta equals 5.

We conclude the following:

For an otherwise average family with two young children and parents who did not attend college, increasing income from \$0 to \$5,000 is expected to decrease the probability of labor force participation by 0.02, which is significant at the 0.05 level but not at the 0.01 level.

If you prefer discrete changes that are centered around the mean, you can add the **centered** option to **mchange**:

```
. mchange inc lwg, atmeans centered
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

	Change	p-value
inc		
+1 centered	-0.009	0.000
+SD centered	-0.099	0.000
Marginal	-0.009	0.000
lwg		
+1 centered	0.148	0.000
+SD centered	0.087	0.000
Marginal	0.149	0.000

Predictions at base value

	not in LF	in LF
Pr(y base)	0.422	0.578

Base values of regressors

	k5	k618	2. agecat	3. agecat	1. wc	1. hc
at	.238	1.35	.385	.219	.282	.392
	lwg	inc				
at	1.1	20.1				

1: Estimates with margins option atmeans.

This can be interpreted as follows:

For an average family, a standard deviation change in income, roughly \$12,000, centered around the mean is expected to decrease the probability of labor force participation by 0.10 ($p < 0.001$).

Because the change is centered, we are computing the effect of changing income from 1/2 standard deviation below the mean of \$20,100 to 1/2 standard deviation above the mean, that is, a change from roughly \$14,300 to \$25,900. In this example, the centered change is nearly identical to the uncentered change with a value of -0.101. Centered and uncentered changes are similar when the change in the independent variable occurs in a region where the probability curve is approximately linear. When the probability curve is changing shape over the region of change, centered and uncentered changes can differ noticeably.

Marginal effects with powers and interactions

As discussed in section 6.2.1, when computing marginal effects for a variable that is linked with other variables, you must ensure that all linked variables change appropri-

ately. You cannot simply change one of the linked variables and assume you can hold the others constant. Fortunately, this is taken care of automatically when linked variables are specified to your model with factor-variable notation.

To show how linked variables are properly handled by `mchange`, we compute MEMs for a standard deviation increase in `inc` and `lwg`. Although we are using MEMs because they make the levels of other variables explicit, which is didactically useful, things work the same way with AMES. We include income and income-squared in the model by including `c.inc c.inc#c.inc` in the command (equivalently, we could have used `c.inc##c.inc`).

```
. logit lfp c.inc c.inc#c.inc lwg k5 k618 i.agecat i.wc i.hc, nolog
Logistic regression                               Number of obs   =       753
                                                    LR chi2(9)      =      128.00
                                                    Prob > chi2     =       0.0000
Log likelihood = -450.87545                        Pseudo R2      =       0.1243
```

	lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
	inc	-.0692521	.0195383	-3.54	0.000	-.1075465	-.0309577
	c.inc#c.inc	.0005123	.0002572	1.99	0.046	8.28e-06	.0010163
(output omitted)							
	_cons	1.381411	.3461711	3.99	0.000	.7029285	2.059894

Estimates of the coefficient for income and income-squared, labeled `c.inc#c.inc`, are shown. Using `mchange`, we compute MEMs for `inc` and `lwg`:

```
. mchange inc lwg, atmeans amount(sd)
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

		Change	p-value				
inc							
	+SD	-0.123	0.000				
lwg							
	+SD	0.087	0.000				
Predictions at base value							
		not in LF	in LF				
	Pr(y base)	0.438	0.562				
Base values of regressors							
		inc	lwg	k5	k618	2. agecat	3. agecat
at		20.1	1.1	.238	1.35	.385	.219
		1. wc	1. hc				
at		.282	.392				

1: Estimates with margins option atmeans.

The base values include the mean of `inc` but do not mention `c.inc#c.inc`. This is because the mean of the variable income-squared is not computed, but instead the mean of income is squared. This is what we want. If we had added income-squared to our model by creating a new variable (for example, `gen incsq = inc*inc`) instead of using `c.inc#c.inc`, the `atmeans` option would have incorrectly held `incsq` at its mean. Again, one of the great advantages of using factor-variable notation is that Stata will automatically handle this correctly.

The discrete change for income in the model is -0.123 . We can interpret the discrete change just as we would in a model that did not include income-squared:

For someone who is average on all characteristics, an increase of one standard deviation in family income is predicted to decrease the probability of being in the labor force by 0.12.

Although we do not provide an example here with interaction terms, the same principle applies. If factor-variable notation is used, `margins` and our `m*` commands will handle the computations of marginal and discrete changes correctly. If you do not use factor-variable notation, you can still get `margins` to produce the right answers, but it requires you to do work that Stata can handle automatically.

6.2.5 The distribution of marginal effects

The value of a marginal effect depends on the level of all variables in the model. Because each observation can have different values of the independent variables, there is a distribution of marginal effects within the sample where the AME is the mean of this distribution. Although the mean tells you where the center of the distribution is, it does not reflect variation within the distribution. Just as the means of the independent variables used to compute the MEM might not correspond even approximately to anyone in the sample, the AME might not correspond to the magnitude of the marginal effect for anyone in the sample. For this reason, we believe that examining the distribution of marginal effects provides valuable substantive insights.

We consider two approaches for learning about the distribution of marginal effects.³ First, we compute effects for each observation and create a histogram of the effects. Although there is no Stata command for this, we provide simple programs that you can adapt to your needs. Second, we compute marginal effects at strategic locations in the data space by using MERs. This approach is presented in section 6.3.

3. A third approach that we do not consider here estimates the quantiles of the effects in the population; see Firpo (2007) and Cattaneo (2010) for seminal papers, and see Cattaneo, Drukker, and Holland (2013) and Drukker (2014) for intuition, Stata commands, and extensions to survival data. For example, a training program that boosts the income of low-income participants and has no effect on higher-income participants could have the 0.25 quantile effect be significant and the 0.75 quantile effect be insignificant. These quantiles of effects provide the researcher with a more nuanced picture of the effect of a treatment than the one provided by the mean effect.

The marginal change for the BRM, assuming no interactions or power terms, equals

$$\frac{\partial \Pr(y_i = 1 \mid \mathbf{x}_i)}{\partial x_k} = f(\mathbf{x}_i\boldsymbol{\beta})\beta_k$$

The shape of the distribution of marginal changes for each observation is determined by $f(\mathbf{x}_i\boldsymbol{\beta})$, where β_k simply rescales $f(\mathbf{x}_i\boldsymbol{\beta})$ to create the distribution of effects for x_k . The distribution is more spread out if β_k is larger in absolute value and is more condensed if β_k is smaller. Although the shape of the distribution of discrete changes will be similar for different variables, they are not a simple rescaling of each other.

There are several ways to compute the marginal changes for each observation. For the logit model, the simplest approach is to use the formula

$$\frac{\partial \Pr(y_i = 1 \mid \mathbf{x}_i)}{\partial x_k} = \Pr(y_i = 1 \mid \mathbf{x}_i) \{1 - \Pr(y_i = 1 \mid \mathbf{x}_i)\} \beta_k$$

where $\Pr(y_i = 1 \mid \mathbf{x}_i) \{1 - \Pr(y_i = 1 \mid \mathbf{x}_i)\}$ is the PDF for the logistic distribution. After `predict` computes $\Pr(y_i = 1 \mid \mathbf{x}_i)$ for each observation, it is easy to create a variable containing the marginal effects:

```
. predict double prhat if e(sample)
(option pr assumed; Pr(lfp))
. gen double mcinc = prhat * (1-prhat) * _b[inc]
. label var mcinc "Marginal change of inc on Pr(LFP)"
```

where `_b[inc]` is the estimated regression coefficient for `inc`. For a probit model, we compute the PDF by using the `normalden()` function:

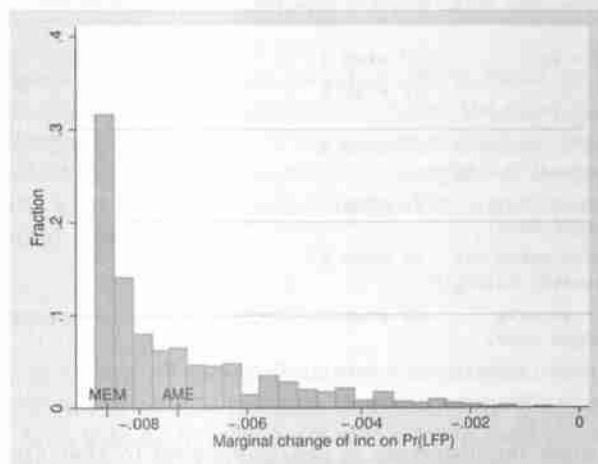
```
. probit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
. predict double pbtxb, xb
. label var pbtxb "xb"
. gen double pbtpdf = normalden(pbtxb)
. label var pbtpdf "normal pdf at xb"
. gen double pbtmcinc = pbtpdf * _b[inc]
. label var pbtmcinc "Marginal change of inc from probit"
```

Next, we plot the distribution of marginal changes with the `histogram` command. To annotate the graph with the values of the AME, we use local macros to hold estimates from `mchange`. After running `mchange inc`, the AME is saved in the third row and first column of the return matrix `r(table)`. To see this, you can enter the command `matlist r(table)` after `mchange`. We place the estimate in the local macro named `ame` by using the `el()` function, which retrieves the value of a single element from a matrix. We do the same thing for the MEM.

```
. quietly mchange inc
. local ame = el(r(table),3,1)
. quietly mchange inc, atmeans
. local mem = el(r(table),3,1)
```


The histogram of marginal changes is created with the following command, using `text()` to add the AME and MEM:

```
. histogram mcinc, xlab(-.008(.002)0) ylab(0(.1).4,grid)
> fraction bin(25) col(gs10) fcol(gs12)
> text(.015 `ame` "AME", place(center))
> text(.000 `ame` "|", place(center))
> text(.015 `mem` "MEM", place(center))
> text(.000 `mem` "|", place(center))
(bin=25, start=-.00876351, width=.00033178)
```



The distribution of marginal changes for income is highly skewed, ranging from -0.10 to less than -0.005 . Because of the skew, the MEM is a better indicator of what we would expect for most respondents than is the AME. The graph shows that over 30% of the sample have effects similar to that of an “average” person. On the other hand, 37% of the sample have effects that are smaller (to the right in the histogram) than the AME.

To compute the distribution of discrete changes for `wc`, we use counterfactual predictions. The same approach could also be used to compute discrete changes for a continuous variable. Before proceeding, a word of caution: This approach to computing effects involves changing the original data that were used to fit the model. *It is essential that you do not save the changed data.*

- Step 1: The original variable `wc` is copied to variable `wc.orig` so that `wc` can be changed.
- Step 2: All cases are assigned `wc=0` to create the counterfactual condition that no women went to college.
- Step 3: Predicted probabilities are computed assuming that no women went to college.
- Step 4: All cases are assigned `wc=1` to create the counterfactual condition that all women went to college.

Step 5: Predicted probabilities are computed assuming that all women went to college.

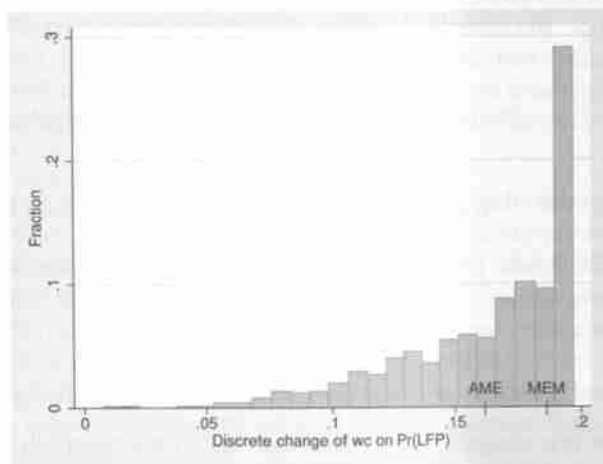
Step 6: The variable `wc` is restored to the original values that were saved in `wc_orig` in Step 1.

Step 7: The difference between the predictions from Step 5 where `wc=1` and Step 3 where `wc=0` is the discrete change for each case; the average of the differences is the average discrete change.

The Stata commands that make these computations are as follows:

```
. gen wc_orig = wc          // step 1
. replace wc = 0            // step 2
(212 real changes made)
. predict double prhat_wc0  // step 3
(option pr assumed; Pr(lfp))
. replace wc = 1            // step 4
(753 real changes made)
. predict double prhat_wc1  // step 5
(option pr assumed; Pr(lfp))
. replace wc = wc_orig      // step 6
(541 real changes made)
. gen double dc_wc = prhat_wc1 - prhat_wc0 // step 7
. label var dc_wc "Discrete change of wc on Pr(LFP)"
```

The command to draw the histogram is nearly identical to that above, so we do not repeat it. The following graph for the discrete change for `wc` is produced:



After plotting the distribution of effects, a useful next step is to determine the marginal effects for ideal types that represent distinct characteristics in the sample. This topic is considered in section 6.3.

6.2.6 (Advanced) Algorithm for computing the distribution of effects

In this section, we use `margins` and slightly more advanced programming techniques to create a general algorithm for plotting the distribution of effects. Although the programming is more complicated, the code works with any model that is compatible with `margins`, even if your model includes interactions and product terms. We suggest you read this section after you have mastered other materials in this chapter.

Instead of using `generate` to compute marginal effects based on the formula for a specific model, this algorithm uses `margins` to compute the effect for each observation. Although this is computationally slow, it works very generally for creating a histogram of any marginal effect that can be computed by `margins` or by predictions made by `margins`. We begin with a review of using `margins` to compute marginal effects (see section 4.5 for related information).

Using `margins` to compute marginal effects

The option `dydx(varname)` tells `margins` to compute marginal effects. If *varname* is a factor variable, such as `i.wc` in our example, `margins` computes the discrete change as *varname* changes from 0 to 1. If *varname* is not a factor variable, `margins` computes the marginal change (that is, partial derivative) for *varname*.

We begin by fitting the model and storing the results. We must store them because we will use the `post` option with `margins`, which replaces the regression estimates in memory with the results from `margins`.

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lvg inc, nolog
(output omitted)
. estimates store mymodel
```

Next, we compute the marginal change with `margins`, `dydx(inc)`, leaving the results in the return `r(b)`.


```
. margins, dydx(wc)
Average marginal effects
Model VCE      : OIM
Expression     : Pr(lfp), predict()
dy/dx w.r.t.   : 1.wc
Number of obs   =      753
```

	Delta-method				[95% Conf. Interval]	
	dy/dx	Std. Err.	z	P> z		
1.wc college	.1624037	.0440211	3.69	0.000	.076124	.2486834

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

```
. matlist r(b)
      Ob.      1.
      wc      wc
y1 |      0      .1624037
```

We can also use `margins` to compute discrete changes for continuous variables, but this takes two steps. First, we make two predictions and post the results. Second, we use `lincom` or `margins` to compute the discrete change. For example, suppose that we want to compute the change in the probability of labor force participation as the number of young children increases from 0 to 3. We compute predictions with two *atspecs*, one for `k5=0` and the other for `k5=3`:

```
. margins, at(k5=0) at(k5=3) post
Predictive margins
Model VCE      : OIM
Expression     : Pr(lfp), predict()
1._at         : k5      =      0
2._at         : k5      =      3
Number of obs   =      753
```

	Delta-method				[95% Conf. Interval]	
	Margin	Std. Err.	z	P> z		
1._at 1	.6370361	.0182192	34.97	0.000	.6013271	.6727452
2._at 2	.0382865	.0182757	2.09	0.036	.0024669	.0741061

```
. matlist e(b)
      1.      2.
      _at      _at
y1 | .6370361 .0382865
```

Because we used the `post` option, the predictions are saved to `e(b)`, which allows us to use `mlincom` (or `lincom`) to compute the average discrete change:


```
. mlincom 2 - 1
```

	lincom	pvalue	ll	ul
1	-0.599	0.000	-0.656	-0.541

The linear combination in the column `lincom` is returned to `r(est)`.

We can also compute a change of a fixed amount from the observed values, for example, the average change as `inc` increases by 1 from its observed values. To do this, we will want to use a single `margins` command to produce two different predictions: one in which predictions are computed at the observed values and one in which `inc` is increased by 1. To get predictions at the observed values, we simply want to specify `at()` with an empty *atspec*. To get predictions in which `inc` is increased by 1, we must use the `gen()` option, added in Stata 13, when specifying the *atspec*. The specification `at(inc=gen(inc+1))` tells `margins` to increase `inc` by 1 from its observed values before computing the prediction. We use these two *atspecs* after we restore the logit results from our base model.

```
. estimates restore mymodel
(results mymodel are active now)
. margins, at() at(inc=gen(inc+1)) post
Predictive margins                                Number of obs =      753
Model VCE    : OIM
Expression   : Pr(lfp), predict()
1._at        : (asobserved)
2._at        : inc          = inc+1
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_at						
1	.5683931	.0166014	34.24	0.000	.535855	.6009312
2	.5611046	.0167439	33.51	0.000	.5282871	.5939221

Because we used the `post` option, we can use `mlincom` to compute the change:

```
. mlincom 2 - 1
```

	lincom	pvalue	ll	ul
1	-0.007	0.000	-0.011	-0.004

The results are identical to those produced by `mchange inc, amount(one)`.

Algorithm for computing the distribution of marginal effects

In this section, we explain how to write a do-file that computes marginal effects for each observation in the estimation sample. The effects are saved in a variable that is plotted. We begin with an overview of the steps involved.

Prepare to compute effects:

Step 1: Fit the regression model.

Step 2: Create the temporary variable ``insample'` from `e(sample)`, indicating which cases are in the estimation sample. (Temporary variables are variables that will automatically be erased when your do-file ends. For more information, type `help tempvar` or see [P] `macro`.)

Step 3: Create the temporary variable ``effect'` to hold marginal effects. This variable is graphed with `histogram` to show the distribution of marginal effects.

Loop through observations and compute effects:

Step 4: Determine whether a case is in the estimation sample by using the temporary variable ``insample'`.

Step 5: Use `margins` to compute the marginal effect for the current case. Any of the methods of computing effects from the prior section can be used.

Step 6: Save the effect for the current case in the corresponding row of the variable ``effect'`.

Verify results and plot effects:

Step 7: Compute the mean of ``effect'` and compare it with the AME computed by `margins`. If these are not the same, there is an error in your program.

Step 8: Plot the distribution of effects by creating a histogram of ``effect'`.

Step 9: If you want to save the effects for each observation, generate a variable equal to the temporary variable ``effect'`.

Using these steps, we compute the marginal effects for `wc`.

```
.. // step 1: estimate the model
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
```



```

. // step 2: create a variable containing e(sample)
. tempvar insample
. gen `insample' = e(sample)
. label var `insample' "In estimation sample?"

. // step 3: create a variable to hold effects
. tempvar effect
. gen `effect' = .
(753 missing values generated)
. label var `effect' "Marginal effect for each observation"

. // loop through all observations
. local nobs = _N
. forvalues i = 1/`nobs' {
2.
.   if `insample'[`i']==1 { // step 4: only cases in estimation sample
3.
.     // step 5: use margins to compute effect for current case
.     qui margins in `i', dydx(wc) nose
4.
.     // step 6: save marginal effect in variable
.     qui replace `effect' = _r(b)_1_2 in `i'
5.   }
6. }

. // step 7: compare average of effect variable to AME from margins
. sum `effect'

Variable | Obs      Mean      Std. Dev.      Min      Max
-----|-----
_000001 |    753    .1624037    .0344572    .0074083    .1968259

. margins, dydx(wc)
Average marginal effects              Number of obs   =        753
Model VCE      : OIM
Expression    : Pr(lfp), predict()
dy/dx w.r.t.  : 1.wc

+-----+-----+-----+-----+-----+-----+
|                Delta-method                |
|      dy/dx      Std. Err.      z    P>|z|      [95% Conf. Interval]
+-----+-----+-----+-----+-----+-----+
|      wc      |
| college |
+-----+-----+-----+-----+-----+-----+
|      .1624037    .0440211     3.69    0.000     .076124     .2486834
+-----+-----+-----+-----+-----+-----+

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

. // step 8: plot the distribution of effects
. histogram `effect', title(Distribution of marginal effects for wc)
(bin=27, start=.00740829, width=.00701547)

```

In this example, `margins` computed the marginal effect by using `dydx()`. The code can be modified so that `margins` computes two predictions (for example, with `wc=0` and with `wc=1`) and the discrete change can be computed with `margins` or `lincom`. For models with multiple outcomes, such as `mlogit` or `oprobit`, the option `predict(outcome())`

can be used. Although this algorithm is very general, it is also slow because `margins` is run once for each observation; every time `margins` is run, it computes effects for every observation.

6.3 Ideal types

An ideal type is a hypothetical observation with substantively illustrative values. A table of probabilities for ideal types of people, countries, cows, or whatever you are studying can quickly summarize the effects of key variables. In our example of labor force participation, we want to examine four ideal types of respondents:

- A young family with lower income, no college education, and young children.
- A young family with college education and young children.
- A middle-aged family with college education and teenage children.
- An older family with college education and adult children.

We find ideal types to be particularly illustrative for interpretation when independent variables are substantially correlated. In the above example, we first consider the contrast between lower income and no college education and higher income and college education, because these indicators of SES covary strongly enough that it is easy to envision them as low- and high-SES prototypes. Across the latter three examples, we construct ideal types reflecting that the age of parents and their children change together.

We use `mtable` to estimate the probabilities for each of these ideal types. To introduce the command and explain some options, we begin with an example that combines two sets of predictions. (See section 4.4 for an introduction to `mtable`.) We then illustrate two approaches for creating a table of ideal types.

For our first ideal type, we define a young, lower-class family as having the values specified as `at(agecat=1 k5=2 k618=0 inc=10 lwg=.75 hc=0 wc=0)`. `lwg` equals the log of the federal minimum wage for 1975, the year the data were collected. We use `mtable` to make predictions, using the `rowname()` option to label the results. The option `ci`, a synonym for `statistics(ci)`, requests confidence intervals along with the predicted probability. Because this is the first step in constructing a table of predictions, we use the `clear` option to remove from memory any prior predictions saved by `mtable`.


```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 1.agecat 1.wc 1.hc lwg inc, nolog
(output omitted)
. mtable, rowname(1 Young low SES young kids) ci clear
> at(agecat=1 k5=2 k618=0 inc=10 lwg=.75 hc=0 wc=0)
Expression: Pr(lfp). predict()
```

	Pr(y)	ll	ul
1 Young low SES young kids	0.159	0.068	0.251

Specified values of covariates

	k5	k618	agecat	wc	hc	lw
Current	2	0	1	0	0	.75
	inc					
Current	10					

We conclude the following:

For a young, lower SES family with two young children, the estimated probability of being in the labor force is 0.16 with a 95% confidence interval from 0.07 to 0.25.

For our next ideal type, we define a young, college-educated family with young children by using `at(agecat==1 k5==2 k618==0 wc==1 hc==1)`, which specifies the values for all the independent variables except `lw` and `inc`. Because we used the `atmeans` option, these variables are set to the means in the estimation sample. To place the new prediction below the prediction from the last `mtable` command, we use the `below` option.

```
. mtable, rowname(2 Young college young kids) ci below
> at(agecat==1 k5==2 k618==0 wc==1 hc==1) atmeans
Expression: Pr(lfp). predict()
```

	Pr(y)	ll	ul
1 Young low SES young kids	0.159	0.068	0.251
2 Young college young kids	0.295	0.156	0.433

Specified values of covariates

	k5	k618	agecat	wc	hc	lwg
Set 1	2	0	1	0	0	.75
Current	2	0	1	1	1	1.1
	inc					
Set 1	10					
Current	20.1					

Below the table of predictions is a table showing the levels of the covariates when the predictions were made. Although you can suppress its display with the `brief` option, we find it useful for knowing exactly how the ideal types were defined. **Set 1** refers to the first predictions in the table, which we numbered as 1. The row **Current** contains values of the `at()` variables from the current or most recent `mtable` command.

We add two more ideal types that show what happens to the probability of being in the labor force as women and children get older. We use `quietly` to suppress output until the last `mtable` command, which displays the complete table.

```
. quietly mtable, rowname(3 Midage college with teens) ci below
> at(agecat==2 k5==0 k618==2 wc==1 hc==1) atmeans
. mtable, rowname(4 Older college with adult kids) ci below
> at(agecat==3 k5==0 k618==0 wc==1 hc==1) atmeans
Expression: Pr(lfp), predict()
```

	Pr(y)	ll	ul
1 Young low SES young kids	0.159	0.068	0.251
2 Young college young kids	0.295	0.156	0.433
3 Midage college with teen	0.760	0.680	0.840
4 Older college with adult	0.653	0.548	0.758

Specified values of covariates

	k5	k618	agecat	wc	hc	lwg
Set 1	2	0	1	0	0	.75
Set 2	2	0	1	1	1	1.1
Set 3	0	2	2	1	1	1.1
Current	0	0	3	1	1	1.1
	inc					
Set 1	10					
Set 2	20.1					
Set 3	20.1					
Current	20.1					

The first two rows allow us to see the big difference in the labor force between the lower and higher SES families that have young children. The mother from the higher SES family has about twice the probability of being in the labor force. The probability for the higher SES mother increases dramatically as her children are no longer young: the chance of labor force participation goes from about 30% to 76%.

It is important to emphasize that predictions we make about ideal types are predictions about a hypothetical observation, not predictions about a subgroup. When we use ideal types, we will specify a particular value for each of the independent variables, either directly or by using `atmeans` to compute global or, as we will show next, local means. This keeps our understanding of what is meant by an ideal type simpler and the interpretations that use them clearer. What we want to avoid in particular is defining an ideal type by specifying values for some independent variables, but then computing average predictions over a set of observations with `asobserved`. That mixes together the concepts of MERS and AMEs and makes interpreting results very confusing. Again, you can think of an ideal type as a hypothetical observation with one prediction and

think of a subgroup as a set of observations with a distribution of predictions that may be averaged or plotted.

6.3.1 Using local means with ideal types

The last three rows of the table were constructed using the `atmeans` option to specify the values of `inc` and `lw` to the sample means. We refer to means based on the entire estimation sample as global means. Although using global means for each ideal type is simple, it often is not realistic. For example, it is reasonable to assume that levels of income and wages would be higher for college-educated respondents than for those who have not attended college and that they would change with age, which is not reflected in the global means.

To address this problem, we can use local means that are defined based on the characteristics specified in the `at()` statements. To do this, we create a selection variable that equals 1 if an observation is part of the group defined by the conditions of an *atspec* and equals 0 otherwise. In other words, a selection variable indicates whether an observation is part of the group defined for the ideal type. To create these variables, we use the `generate` command with `if` conditions that correspond to the *atspecs* used for an ideal type:

```
. gen _selYC = agecat==1 & k5==2 & k618==0 & wc==1 & hc==1
. label var _selYC "Select Young college young kids"
. gen _selMC = agecat==2 & k5==0 & k618==2 & wc==1 & hc==1
. label var _selMC "Select Midage college with teens"
. gen _selOC = agecat==3 & k5==0 & k618==0 & wc==1 & hc==1
. label var _selOC "Select Older college with adult kids"
```

Once these variables are created, we can make a table of predictions containing local means for variables not explicitly set by the *atspec*. The first row of the table is unchanged from before because all variables for that ideal type were explicitly specified in the `at()` option:

```
. quietly mtable, rowname(1 Young low SES young kids) ci clear
> at(agecat=1 k5=2 k618=0 inc=10 lw=.75 hc=0 wc=0)
```

In the next command, we add `if _selYC==1` to the `mtable` command so that predictions are based only on observations defined by `_selYC`:

```
. quietly mtable if _selYC==1, rowname(2 Young college young kids)
> atmeans ci below
```

The `if` condition selects observations where `agecat==1 & k5==2 & k618==0 & wc==1 & hc==1`, which define our ideal type. The means of these variables will equal their specified values (for example, `agecat` will equal 1 and `k5` will equal 2), while those variables not used to define the selection variable will equal the local mean defined by selection variables. For example, `lw` will equal the average log of wages for young families with college education. Accordingly, the `if` condition makes it easy to specify

the values we wanted to define our ideal type. In the same way, we add the last two ideal types to the table:

```
. quietly mtable if _selMC==1, rowname(3 Midage college with teens)
> atmeans ci below
. mtable if _selOC==1, rowname(4 Older college with adult kids)
> atmeans ci below
```

Expression: Pr(lfp), predict()

	Pr(y)	ll	ul
1 Young low SES young kids	0.159	0.068	0.251
2 Young college young kids	0.394	0.234	0.554
3 Midage college with teen	0.739	0.659	0.820
4 Older college with adult	0.631	0.528	0.734

Specified values of covariates

	k5	k618	agecat	wc	hc	lvg
Set 1	2	0	1	0	0	.75
Set 2	2	0	1	1	1	1.62
Set 3	0	2	2	1	1	1.16
Current	0	0	3	1	1	1.38
<hr/>						
	inc					
Set 1	10					
Set 2	16.6					
Set 3	24.4					
Current	27.9					

An advantage of using local means with ideal types is that the values of variables not specified in the type are held to values more consistent with what is actually observed, so the ideal type more accurately resembles the actual cases in our dataset that share the key features of the ideal type.

6.3.2 Comparing ideal types with statistical tests

The predicted probabilities of labor force participation vary among the four ideal types. Before concluding, for example, that the probability of being in the labor force is greater for a young, college-educated family with children than for a family with no college education, we need to test whether the predictions are significantly different. Essentially, this involves testing whether a discrete change is 0 when the starting values and ending values vary on multiple variables. To show how this is done, we compute two ideal types in the same `mtable` command and post the results so that we can evaluate them with `mlincom`. Because we are posting the results, we begin with `estimates store` so that we can later restore the estimation results from `logit` after they have been replaced by the posted predictions.


```

. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. estimates store base
. mtable, atmeans post
> at(agecat=1 k5=2 k618=0 wc=0 hc=0 lwg=.75 inc=10) // ideal type 1
> at(agecat=1 k5=2 k618=0 wc=1 hc=1 lwg=1.62 inc=16.64) // ideal type 2
Expression: Pr(lfp), predict()

```

	wc	hc	lwg	inc	Pr(y)
1	0	0	.75	10	0.159
2	1	1	1.62	16.6	0.394

Specified values of covariates

	k5	k618	agecat
Current	2	0	1

Now, we estimate the difference in the predictions and end by restoring the estimation results from logit:

```

. mlincom 1 - 2

```

	lincom	pvalue	ll	ul
1	-0.234	0.000	-0.340	-0.129

```

. estimates restore base
(results base are active now)

```

We conclude the following:

A wife from a young, lower SES family with young children is significantly less likely to be in the labor force than a wife from a young family with college education ($p < 0.001$).

6.3.3 (Advanced) Using macros to test differences between ideal types

In this section, we discuss using local macros and returns to automate the process of computing predictions at multiple fixed values of the `at()` variables. If you rarely test the equality of predictions, the methods from the last section should meet your needs. If you often test the equality of predictions, this section can save you time.

It is tedious and error-prone to specify the *atspecs* for multiple ideal types to test the equality of predictions. To automate this process, we can use the returned results from `mtable`. When `mtable` is run with a single `at()`, it returns the local `r(atspec)` as a string that contains the specified values of the covariates. This is easiest to understand with an example:


```
. mtable, atmeans at(agecat=1 k5=2 k618=0 wc=0 hc=0 lwg=.75 inc=10)
```

```
Expression: Pr(lfp), predict()
```

Pr(y)						
0.159						
Specified values of covariates						
	k5	k618	agecat	wc	hc	lwg
Current	2	0	1	0	0	.75
	inc					
Current	10					

The values shown in the Specified values of covariates table are saved in the return `r(atspec)`:

```
. display "`r(atspec)'"
k5=2 k618=0 1b.agecat=1 2.agecat=0 3.agecat=0 0b.wc=1 1.wc=0 0b.hc=1 1.hc=0 lwg=
> .75 inc=10
```

We create a local macro that is used to specify the *atspec* for *mtable*:

```
. local myatspec `r(atspec)'
. mtable, atmeans at(`myatspec')
```

```
Expression: Pr(lfp), predict()
```

Pr(y)						
0.159						
Specified values of covariates						
	k5	k618	agecat	wc	hc	lwg
Current	2	0	1	0	0	.75
	inc					
Current	10					

The results match those we obtained earlier.

Using this strategy and the selection variables created before (see page 273), we create local macros with the *atspecs* for our four ideal types:

```
. quietly mtable, atmeans at(agecat=1 k5=2 k618=0 inc=10 lwg=.75 hc=0 wc=0)
. local YngLow `r(atspec)'

. quietly mtable if _selYC == 1, atmeans
. local YngCol `r(atspec)'

. quietly mtable if _selMC == 1, atmeans
. local MidCol `r(atspec)'

. quietly mtable if _selOC == 1, atmeans
. local OldCol `r(atspec)'
```


We use these locals to compute four predictions with a single `mtable`:

```
. mtable, at(`YngLow`) at(`YngCol`) at(`MidCol`) at(`OldCol`) post
Expression: Pr(lfp), predict()
```

	k5	k618	agecat	wc	hc	lwg
1	2	0	1	0	0	.75
2	2	0	1	1	1	1.62
3	0	2	2	1	1	1.16
4	0	0	3	1	1	1.38

	inc	Pr(y)
1	10	0.159
2	16.6	0.394
3	24.4	0.739
4	27.9	0.631

Specified values where .n indicates no values specified with at()

	No at()
Current	.n

Because the values of all independent variables were specified for each prediction, their values appear in the table of predictions rather than in a table of values of covariates below the predictions. Because there are no values to place in the table, `.n` is shown. Because the predictions were posted, we can use `mlincom` for each comparison:

```
. mlincom 1 - 2
```

	lincom	pvalue	ll	ul
1	-0.235	0.000	-0.340	-0.129

```
. mlincom 1 - 3
```

	lincom	pvalue	ll	ul
1	-0.580	0.000	-0.720	-0.440

```
. mlincom 1 - 4
(output omitted)
```

Alternatively, we can take advantage of the `pwcompare()` option in `margins`, which is not available with the `mtable` command. We specify the values at which we want to make predictions, just like we did with `mtable`. We suppress the lengthy listing of the *atlegend* and request pairwise comparisons of the estimates:


```

, estimates restore base
(results base are active now)
. margins, at(`YngLow`) at(`YngCol`) at(`MidCol`) at(`OldCol`)
>      noatlegend pvcompare(effects)

Pairwise comparisons of adjusted predictions
Model VCE      : OIM
Expression     : Pr(lfp), predict()

```

	Delta-method		Unadjusted		Unadjusted	
	Contrast	Std. Err.	z	P> z	[95% Conf. Interval]	
_at						
2 vs 1	.2346236	.0536979	4.37	0.000	.1293776	.3398696
3 vs 1	.5798713	.0715922	8.10	0.000	.4395531	.7201895
4 vs 1	.4713628	.0771112	6.11	0.000	.3202276	.6224979
3 vs 2	.3452477	.0875971	3.94	0.000	.1735606	.5169349
4 vs 2	.2367392	.0874085	2.71	0.007	.0654216	.4080568
4 vs 3	-.1085085	.0487212	-2.23	0.026	-.2040002	-.0130168

Row 2 vs 1 shows that the increase in predicted probability for the young, college-educated family versus the young, low SES family is significant. Indeed, all differences are significant at the 0.001 level, except for the differences between an older family with adult children and a middle-aged family, which is significant at the 0.05 level.

6.3.4 Marginal effects for ideal types

Given our cautions about relying on a single value to summarize marginal effects, ideal types are an excellent way to examine variation in the size of effects at different locations in the data space. Here we are taking different hypothetical observations and describing how the predicted probability changes as the value of one of the independent variables for that observation changes. To do this, we use local macros to specify the values at which the change is to be computed, just as we did with `mtable`.

First, we compute discrete changes for `wc` and `k5` for a young, low SES family:

```
. mchange wc k5, atmeans amount(one) at(`YngLow`)
logit: Changes in Pr(y) | Number of obs = 753
Expression: Pr(lfp), predict(pr)
```

	Change	p-value
wc		
college vs no	0.137	0.008
k5		
+1	-0.114	0.000

Predictions at base value

	not in LF	in LF
Pr(y base)	0.841	0.159

Base values of regressors

	k5	k618	agecat	wc	hc	lwg
at	2	0	1	0	0	.75
	inc					
at	10					

1: Estimates with margins option `atmeans`.

```
. matrix YngLow = r(table)
```

`mchange` leaves the marginal effects in the `r(table)` matrix, which we copy to the matrix `YngLow` so that we can combine it with estimates of effects for other ideal types:

```
. mchange wc k5, atmeans amount(one) at(`YngCol`)
(output omitted)
. matrix YngCol = r(table)
. mchange wc k5, atmeans amount(one) at(`MidCol`)
(output omitted)
. matrix MidCol = r(table)
. mchange wc k5, atmeans amount(one) at(`OldCol`)
(output omitted)
. matrix OldCol = r(table)
```

Next, we select the first column of each matrix, which contains the effects, and concatenate them into a single matrix we name `me`:


```
. matrix me = YngLow[1...,1], YngCol[1...,1], MidCol[1...,1], OldCol[1...,1]
. matrix colnames me = YngLow YngCol MidCol OldCol
. matlist me, format(%9.2f) twidth(15)
```

		YngLow	YngCol	MidCol	OldCol
wc					
college vs no		0.14	0.17	0.18	0.20
k5					
	+1	-0.11	-0.25	-0.33	-0.33

The effects of the wife going to college are within 0.06 across the four ideal types. The effects of having one more young child in the family, however, increase in magnitude from -0.11 for young families without college education to -0.33 for older families that attended college. The differences in discrete changes for **k5** reflect the variation in the size of effects within the sample.

6.4 Tables of predicted probabilities

When you are interested in the effects of one or more categorical independent variables, a table of predictions can be very effective. For example, our analysis thus far highlights the importance of attending college and having young children. To see how these variables jointly affect the probability of being in the labor force, we can use a simple `mtable` command:

```
. mtable, at(wc=(0 1) k5=(0 1 2 3)) atmeans
Expression: Pr(lfp), predict()
```

	k5	wc	Pr(y)
1	0	0	0.604
2	0	1	0.772
3	1	0	0.275
4	1	1	0.457
5	2	0	0.086
6	2	1	0.173
7	3	0	0.023
8	3	1	0.049

Specified values of covariates

	k618	2. agecat	3. agecat	1. hc	lw	inc
Current	1.35	.385	.219	.392	1.1	20.1

Although this is the information we want, it is not an effective table. We can improve it by using two `at()`'s along with `atvars(wc k5)` to list values of `wc` in the first column followed by values of `k5`. The option `names(columns)` removes the row numbers (see `help matlist` for details on the `names()` option).


```
. mtable, at(wc=0 k5=(0 1 2 3)) at(wc=1 k5=(0 1 2 3)) atmeans
>      atvars(wc k5) names(columns)
Expression: Pr(lfp), predict()
```

1. wc	k5	Pr(y)
0	0	0.604
0	1	0.275
0	2	0.086
0	3	0.023
1	0	0.772
1	1	0.457
1	2	0.173
1	3	0.049

Specified values of covariates

	k618	2. agecat	3. agecat	1. hc	lwg	inc
Current	1.35	.385	.219	.392	1.1	20.1

The table shows the strong effect of education and how the size of the effects differ by the number of young children, but the information still is not presented well.

Our next step is to compute the discrete change for college education conditional on the number of young children:

$$\frac{\Delta \Pr(y = 1 \mid \mathbf{x}, k5)}{\Delta wc (0 \rightarrow 1)}$$

Because `wc` was entered into the model as a factor variable, we can compute the discrete change by using `dydx(wc)`. In the process, let's create an even more effective table that gets close to what we might include in a paper. First, we compute the predictions for `wc=0`:

```
. quietly mtable, estname(NoCol) at(wc=0 k5=(0 1 2 3)) atmeans brief
```

Next, we make predictions for `wc=1`. We use the `right` option to place the predictions to the right of those from the prior `mtable` command, and we use `atvars(_none)` because we do not want the column with `k5` included again:

```
. quietly mtable, estname(College) at(wc=1 k5=(0 1 2 3)) atmeans
>      atvars(_none) right
```

Now, we use the `dydx(wc)` option to compute discrete changes. We place these along with the *p*-value for testing whether the change is 0 to the right:


```
. mtable, estname(Change) dydx(wc) at(k5=(0 1 2 3)) atmeans
> atvars(_none) right stats(estimate p) names(columns) brief
Expression: Pr(lfp), predict()
```

k5	NoCol	College	Change	p
0	0.604	0.772	0.168	0.000
1	0.275	0.457	0.182	0.001
2	0.086	0.173	0.087	0.013
3	0.023	0.049	0.027	0.085

We can summarize these findings:

For someone who is average on all characteristics and has no young children, having attended college significantly increases the predicted probability of being in the labor force by 0.17. The size of the effect decreases with the number of young children. For example, for someone with two young children, the increase is only 0.09, which is significant at the 0.01 level.

Although this table shows clearly how education and children affect labor force participation, it assumes that it is reasonable to change *wc* and *k5* while holding other variables at their global means. This is unrealistic. For example, it is likely that women with three young children will be in the youngest age group, while few people with three young children will be over 50. Each cell in the table represents a different ideal type, but some of the ideal types are substantively unusual, limiting their usefulness as a point of comparison.

We could approach the problem in at least a couple of different ways that we regard as preferable to using global means. Here we consider two approaches. First, we define ideal types in terms of someone from the youngest age category, ages 30–39, for whom having three young children is not substantively unrealistic. Means of the control variables are based on sample members in the youngest age group. Second, we use local means defined by levels of *k5* when making predictions. Thus the means for those with no children differ from those with one child.

Our first approach is to focus on those in the youngest age category. We could do this by adding *agecat=1* to the *atspecs* used above, but, substantively, we think it makes more sense to use the mean values of the other independent variables for someone in this age group. To do this, we specify *if agecat==1* when we use *atmeans* to make our predictions:


```
. mtable if agecat==1, estname(Change) dydx(wc) at(k5=(0 1 2 3)) atmeans
> atvars(k5) stats(estimate p) names(columns)
```

Expression: Pr(lfp), predict()

k5	Change	p
0	0.131	0.000
1	0.197	0.000
2	0.124	0.005
3	0.043	0.058

Specified values of covariates

	k618	agecat	1. wc	1. hc	lwg	inc
Current	1.85	1	.312	.463	1.06	18.8

Notice that `agecat` is 1 in the table of specified values because we are restricting the analyses to those in this age group. Adapting the commands used above, we create a new table illustrating the effects of children and education on labor force participation:

```
. quietly mtable if agecat==1, estname(NoCol) at(wc=0 k5=(0 1 2 3)) atmeans
. quietly mtable if agecat==1, estname(College) at(wc=1 k5=(0 1 2 3)) atmeans
> atvars(_none) right
```

```
. mtable if agecat==1, estname(Change) dydx(wc) at(k5=(0 1 2 3)) atmeans
> atvars(_none) right stats(estimate p) names(columns) brief
```

Expression: Pr(lfp), predict()

k5	NoCol	College	Change	p
0	0.720	0.851	0.131	0.000
1	0.390	0.586	0.197	0.000
2	0.137	0.261	0.124	0.005
3	0.038	0.081	0.043	0.058

The predicted probabilities and discrete changes using local means for the youngest members of the sample are different from when we used global means for the sample as a whole. Using global means, having a college education increased the predicted probability for a woman with no children by 0.168, while using means conditional on the woman being age 30–39, the increase is 0.131.

Our second approach uses local means defined by levels of `k5` when making predictions. That is, when making predictions for women with no young children, we will hold other variables at the mean for those with no young children, which would be equivalent to computing the means by using `summarize ... if k5==0`. We can do this with the `over(overvar)` option, which makes predictions at each value of `overvar`, where this variable must have nonnegative integer values. For each value of `k5`, predictions are made based only on cases with the given value of `k5`. That is, predictions are made first with cases that meet the condition `k5==0`, then with cases that meet `k5==1`, and so on. Accordingly, the `atmeans` option will hold other variables at the means conditional on the value of `k5`. The output shows how the means vary:


```
. mtable, at(wc=(0 1)) atmeans over(k5) brief
Expression: Pr(lfp), predict()
```

	k5	k618	2. agecat	3. agecat	wc	1. hc
1	0	1.28	.436	.269	0	.358
2	1	1.75	.212	.0169	0	.517
3	2	1.31	.0385	0	0	.538
4	3	1.33	0	0	0	1
5	0	1.28	.436	.269	1	.358
6	1	1.75	.212	.0169	1	.517
7	2	1.31	.0385	0	1	.538
8	3	1.33	0	0	1	1

	lwg	inc	Pr(y)
1	1.11	20	0.583
2	1.03	20.8	0.337
3	1.18	17.6	0.154
4	1.08	46.1	0.017
5	1.11	20	0.757
6	1.03	20.8	0.530
7	1.18	17.6	0.288
8	1.08	46.1	0.037

In row 1 where $wc=0$ and row 5 where $wc=1$, the values for $k618$, $agecat$, wc , hc , lwg , and inc are the means in the subsample defined by $k5=0$. And so on for other values of $k5$. Using `over()` here is equivalent to running a series of `mtable` commands of the form `mtable if k5==0, at(wc=(0 1)) atmeans`.

Does using local means affect the results? In this example, the results using global means do differ somewhat from those using local means, especially at $k5=2$.

k5	Global			Local			Global - Local		
	NoCol	Col	Chng	NoCol	Col	Chng	NoCol	Col	Chng
0.00	0.60	0.77	0.17	0.58	0.76	0.17	-0.02	-0.02	0.01
1.00	0.27	0.46	0.18	0.34	0.53	0.19	0.06	0.07	0.01
2.00	0.09	0.17	0.09	0.15	0.29	0.13	0.07	0.11	0.05
3.00	0.02	0.05	0.03	0.02	0.04	0.02	-0.01	-0.01	-0.01

Tables of predictions allow readers to see how predictions vary over values of substantively important independent variables. We do not want a change in an independent variable to result in our presenting as illustrative some ideal types that are actually unlikely or even impossible. As we have shown above, one way to do this is to restrict our sample or otherwise choose representative values so that the changes presented in the table remain substantively realistic. Another is to use local means that change as the key independent variable(s) of the table changes.

6.5 Second differences comparing marginal effects

We can compute AMEs based on a subset of observations. For example, suppose that we are interested in ways in which the wife's and the husband's educations interact to affect labor force participation. Because we are focusing on the joint effects of these two variables, we fit a new model that includes the interaction between *wc* and *hc*:

```
. logit lfp k5 k618 i.agecat wc#hc lwg inc, nolog
Logistic regression               Number of obs   =       753
                                LR chi2(9)         =      125.57
                                Prob > chi2        =       0.0000
                                Pseudo R2          =       0.1219
Log likelihood = -452.08908
```

lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
(output omitted)						
wc						
college	1.194263	.4344336	2.75	0.006	.3427885	2.045737
hc						
college	.2467267	.2287125	1.08	0.281	-.2015415	.6949949
wc#hc						
college #						
college	-.5586704	.5051034	-1.11	0.269	-1.548655	.431314
(output omitted)						

We want to know whether the effect of a women going to college is the same for a women whose husband did go to college as for a woman whose husband did not go to college:

$$H_0: \frac{\Delta \Pr(y = 1 \mid \mathbf{x}, hc = 0)}{\Delta wc} = \frac{\Delta \Pr(y = 1 \mid \mathbf{x}, hc = 1)}{\Delta wc}$$

To test this hypothesis, we compute the AME of *wc* averaging over only those cases where *hc* is 0 and compare it with the AME for those cases where *hc* is 1. Although we could compute these discrete changes by using `mchange wc if hc==1` and `mchange wc if hc==0`, this will not allow us to test whether the effects are equal because the estimates cannot be posted for testing with `mlincom`. To test the hypothesis, we use `mtable` with the `dydx(wc)` option to compute the discrete change for *wc* and the `over(hc)` option to request the changes be computed with the subgroups defined by *hc*:


```
. mtable, dydx(wc) over(hc) stat(ci) post
```

```
Expression: Pr(lfp), predict()
```

	d Pr(y)	ll	ul
no	0.233	0.094	0.373
college	0.128	0.022	0.234

Specified values where .n indicates no values specified with at()

	No at()
Current	.n

The row labeled **no** contains the discrete change of *wc* for those women whose husbands did not attend college (**no** is the value label for *hc* = 0), and the row **college** contains the change for those whose husbands attended college. The **post** option saves the estimates to **e(b)**, which allows us to use **mlincom** to test whether the marginal effects are equal:

```
. mlincom 1 - 2
```

	lincom	pvalue	ll	ul
1	0.105	0.233	-0.068	0.279

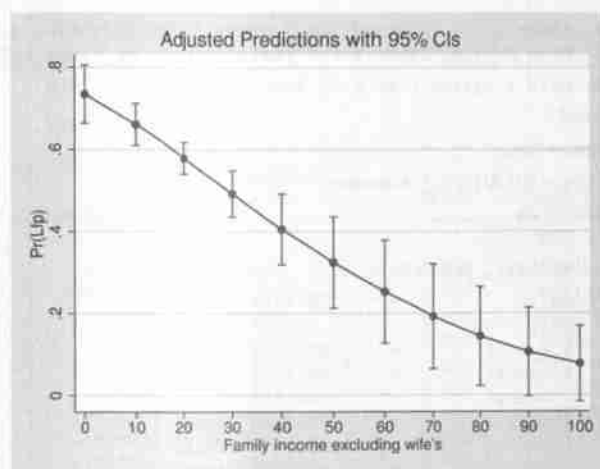
We conclude the following:

Although the average effect of the wife going to college is 0.10 larger when the husband did not go to college than when he did, this difference is not significant ($p > 0.10$).

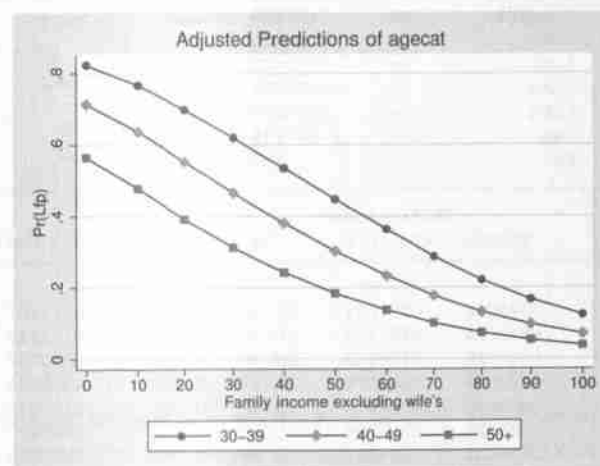
6.6 Graphing predicted probabilities

With a continuous independent variable, you can plot the predicted probabilities over the range of the variable. For example, to examine the effects of *inc*, we might plot the predicted probability of labor force participation as *inc* changes, holding other variables at fixed values. We offer two approaches for making such graphs. First, Stata's **marginsplot** command uses predictions from **margins** to create plots. As you will see, it quickly produces effective graphs. The second approach uses our **mgen** command to generate variables with the values to be plotted, which are then plotted with **graph**. This is essentially what **marginsplot** does behind the scenes. Although **marginsplot** is simpler, **mgen** is more flexible in ways that often justify the greater effort that it requires. The advantage will be particularly apparent in subsequent chapters when we create plots for multiple outcomes that cannot be created with **marginsplot**.

We begin by showing you how to create plots where one variable changes while all other variables are held constant, such as this one:



Next, we show how to introduce the effects of other variables by plotting multiple lines that correspond to different levels of one or more of the variables in the model. For example, the following graph shows the effect of income for respondents with different ages:



6.6.1 Using marginsplot

The first step is to use `margins` to compute predicted probabilities as income increases from 0 to 100, while holding other variables at their means:


```

. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
(output omitted)
. estimates store base
. margins, at(inc=(0(10)100)) atmeans
Adjusted predictions                                Number of obs   =       753
Model VCE      : OIM
Expression     : Pr(lfp), predict()
1._at          : k5                                =       .2377158 (mean)
                  k618                             =       1.353254 (mean)
                  1.agecat                           =       .3957503 (mean)
                  2.agecat                           =       .3851262 (mean)
                  3.agecat                           =       .2191235 (mean)
                  0.wc                                =       .7184595 (mean)
                  1.wc                                =       .2815405 (mean)
                  0.hc                                =       .6082337 (mean)
                  1.hc                                =       .3917663 (mean)
                  lwg                                  =       1.097115 (mean)
                  inc                                  =              0
(output omitted)
11._at          : k5                                =       .2377158 (mean)
                  k618                             =       1.353254 (mean)
                  1.agecat                           =       .3957503 (mean)
                  2.agecat                           =       .3851262 (mean)
                  3.agecat                           =       .2191235 (mean)
                  0.wc                                =       .7184595 (mean)
                  1.wc                                =       .2815405 (mean)
                  0.hc                                =       .6082337 (mean)
                  1.hc                                =       .3917663 (mean)
                  lwg                                  =       1.097115 (mean)
                  inc                                  =          100

```

	Delta-method					[95% Conf. Interval]
	Margin	Std. Err.	z	P> z		
_at						
1	.7349035	.0361031	20.36	0.000	.6641427	.8056643
2	.6613024	.0261131	25.32	0.000	.6101217	.7124832
3	.5789738	.0196943	29.40	0.000	.5403737	.6175739
4	.4920058	.0286579	17.17	0.000	.4358374	.5481742
5	.405519	.0440915	9.20	0.000	.3191012	.4919367
6	.324523	.0569264	5.70	0.000	.2129492	.4360968
7	.2528245	.064092	3.94	0.000	.1272066	.3784425
8	.1924535	.0652874	2.95	0.003	.0644926	.3204144
9	.1437253	.06172	2.33	0.020	.0227563	.2646942
10	.1057196	.0551469	1.92	0.055	-.0023663	.2138055
11	.0768617	.0472071	1.63	0.103	-.0156524	.1693858

The *atlegend* shows the values of the independent variables for each of the 11 predictions in the table, which are automatically saved in the matrix *r(b)*. Because the *atlegend* can be quite long, we often use *noatlegend* to suppress it. Then, we use *mlstat* for a more compact summary.


```
. margins, at(inc=(0(10)100)) atmeans noatlegend
```

```
Adjusted predictions
```

```
Number of obs = 753
```

```
Model VCE : OIM
```

```
Expression : Pr(1fp), predict()
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_at						
1	.7349035	.0361031	20.36	0.000	.6641427	.8056643
2	.6613024	.0261131	25.32	0.000	.6101217	.7124832
3	.5789738	.0196943	29.40	0.000	.5403737	.6175739
4	.4920058	.0286579	17.17	0.000	.4358374	.5481742
5	.405519	.0440915	9.20	0.000	.3191012	.4919367
6	.324523	.0569264	5.70	0.000	.2129492	.4360968
7	.2528245	.064092	3.94	0.000	.1272066	.3784425
8	.1924535	.0652874	2.95	0.003	.0644926	.3204144
9	.1437253	.06172	2.33	0.020	.0227563	.2646942
10	.1057196	.0551469	1.92	0.055	-.0023663	.2138055
11	.0768617	.0472071	1.63	0.103	-.0156624	.1693858

```
. mlistat
```

```
at() values held constant
```

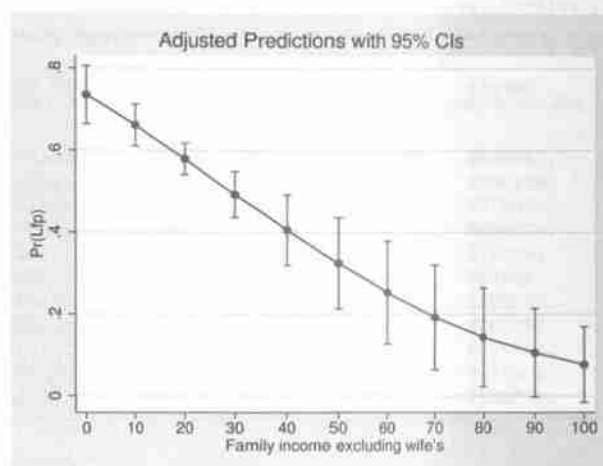
k5	k618	2. agecat	3. agecat	1. wc	1. hc	lw
.238	1.35	.385	.219	.282	.392	1.1

```
at() values vary
```

_at	inc
1	0
2	10
3	20
4	30
5	40
6	50
7	60
8	70
9	80
10	90
11	100

Either way, `marginsplot` uses the predictions in `r(b)` along with other returns from `margins`, and it graphs the predictions including the 95% confidence intervals:


```
. marginsplot
Variables that uniquely identify margins: inc
```



The graph shows how the probability of being in the labor force decreases with family income. It also shows that the confidence intervals are smaller near the center of the data (the mean of `inc` is 20.1) and increase as we move to the extremes.

Although `marginsplot` does an excellent job of creating the graph without requiring options, you can fully customize the graph. Use `help marginsplot` for full details. For example, to suppress the confidence interval, type `marginsplot, noci`. To use shading to show the confidence interval (illustrated below), type `marginsplot, recast(line) recastci(rarea)`.

If you are only interested in plotting a single type of prediction from one model, there is little reason to use anything but `marginsplot`. But, if you want to plot multiple outcomes, such as for multinomial logit, or predictions for a single outcome from multiple models, it is worth learning about `mgen`.

6.6.2 Using `mgen` with the `graph` command

To create the same graph as above by using `mgen`, our first step is to generate variables for plotting:


```
. mgen, atmeans at(inc=(0(10)100)) stub(PLT) predlabel(Pr(LFP))
Predictions from: margins, atmeans at(inc=(0(10)100)) predict(pr)
Variable   Obs Unique   Mean   Min   Max   Label
-----
PLTpr1     11     11   .3608011 .0768617 .7349035 Pr(LFP)
PLTl11     11     11   .2708139 -.0156624 .6641427 95% lower limit
PLTu11     11     11   .4507883 .1693859 .8056643 95% upper limit
PLTinc     11     11     50      0     100 Family income excluding...
```

Specified values of covariates

k5	k618	2. agecat	3. agecat	1. wc	1. hc	lw
.2377158	1.353254	.3851262	.2191235	.2815405	.3917663	1.097115

The option `stub()` specifies the prefix for variables that are generated. If `stub()` is not specified, the default `stub(_)` is used. If you want to replace existing plot variables (perhaps while debugging your *do-file*), add the option `replace`. The option `predlabel()` customizes the variable label for `PLTpr1`, which is handy because by default *graph* uses this label for the *y* axis.

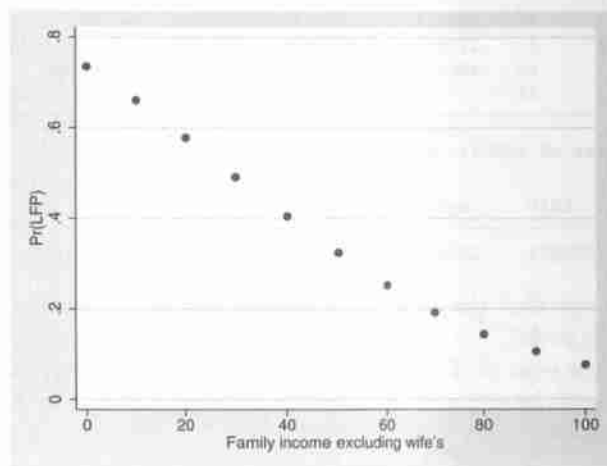
If we list the values for the first 13 observations, we see the variables created by *mgen*:

```
. list PLTinc PLTpr PLTl1 PLTu1 lfp in 1/13, clean
      PLTinc  PLTpr  PLTl1  PLTu1  lfp
1.      0   .7349035 .6641427 .8056643 not in LF
2.     10   .6613024 .6101217 .7124832 not in LF
3.     20   .5789738 .5403737 .6175739 not in LF
4.     30   .4920058 .4358374 .5481742 not in LF
5.     40   .4055189 .3191012 .4919367 not in LF
6.     50   .324523 .2129492 .4360968 not in LF
7.     60   .2528245 .1272066 .3784425 not in LF
8.     70   .1924535 .0644926 .3204144 not in LF
9.     80   .1437253 .0227563 .2646942 not in LF
10.    90   .1057196 -.0023663 .2138055 not in LF
11.   100   .0768617 -.0156624 .1693859 not in LF
12.    .      .      .      .      not in LF
13.    .      .      .      .      not in LF
```

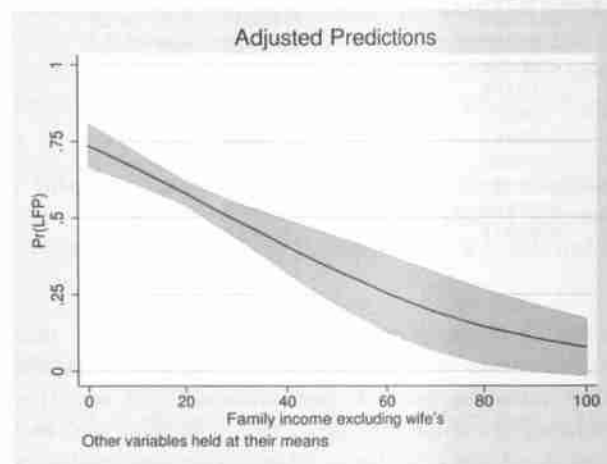
Column 1 contains the 11 values of income from variable `PLTinc` that will define the *x* coordinates. The next column contains predicted probabilities computed at the values of income with other variables held at their means. The negative effect of income is shown by the increasingly small probabilities. The next two columns contain the upper and lower bounds of the confidence intervals for the predictions. The first four variables have missing values beginning in rows 12 and 13 because our *atspec* requested only 11 predictions. The last column shows the observed variable `lfp`, which does not have missing values. This is being shown to remind you that the variables created for graphing are added to the dataset used for estimation.

You can also create a basic graph without confidence intervals:

```
. scatter PLTPr PLTinc
```



Next, we want to add the 95% confidence interval around the predictions. This requires more complicated `graph` options. To explain these, let's start by looking at the graph we want to create:



Here is the `twoway` command that we will explain:

```
. twoway
> (rarea PLTl1 PLTl1 PLTinc, color(gs12))
> (connected PLTPr PLTinc, msymbol(i))
> , title("Adjusted Predictions")
> caption("Other variables held at their means")
> ytitle(Pr(LFP)) ylabel(0(.25)1, grid gmin gmax) legend(off)
```


The first thing to realize is that the `twoway` command includes two plots, a `rarea` plot and a `connected` plot. These are overlaid to make a single graph. First, the shaded confidence intervals are created with a `rarea` plot where the area on the y axis is shaded between the values of `PLTu1` for the upper level or bound and `PLTl1` for the lower bound. We chose `color(gs12)` to make the shading gray scale level 12, a matter of personal preference. Second, the line with predicted probabilities is created with a `connected` plot, where `msymbol(i)` specifies that the symbols (shown as solid circles in our prior graph) that are connected should be invisible—that is, draw the line without symbols. We defined the `rarea` plot before the `connected` plot because Stata draws overlaid plots in the order specified; we want the line indicating the predicted probabilities to appear on top of the shading.

After the “,” in the fourth line of the command are options that apply to the overall graph rather than the individual plots. `ylabel()` defines the y -axis labels, with `grid` requesting grid lines. Suboptions `gmin` and `gmax` place grid lines at the maximum and minimum values of the axis. By default, when you are plotting multiple outcomes—in this case `PLTu1`, `PLTl1`, and `PLTpr`—`graph` adds a legend describing each outcome. To turn this off, we use `legend(off)`. See section 2.17 for more information on graphing.

6.6.3 Graphing multiple predictions

An effective way to show the effects of two variables is to graph predictions at various levels of one variable as the other variable changes. This can be done with either `marginsplot` or `mgen`.

Using `marginsplot`

We can plot the effects of income for each of the age groups. First, we compute the predictions with `margins`, where `margins agecat` indicates that we want predictions for each level of the factor variable `agecat`. `at(inc=(0(10)100)) atmeans` specifies predictions as `inc` increases from 0 to 100 by 10s, with all variables except `agecat` held at their means:


```
. margins agecat, at(inc=(0(10)100)) atmeans noatlegend
```

```
Adjusted predictions
```

```
Number of obs = 753
```

```
Model VCE : OIM
```

```
Expression : Pr(lfp), predict()
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_at#agecat						
1#30-39	.8236541	.031251	26.36	0.000	.7624033	.8849049
1#40-49	.7139289	.0452033	15.79	0.000	.6253321	.8025256
1#50+	.5651833	.0614528	9.20	0.000	.4447381	.6856285
2#30-39	.7668772	.0298035	25.73	0.000	.7084634	.825291
2#40-49	.6373779	.0374018	17.04	0.000	.5640716	.7106841
2#50+	.4779353	.050722	9.42	0.000	.3785219	.5773487
3#30-39	.6985115	.0319211	21.88	0.000	.6359474	.7610756
3#40-49	.5531632	.0322948	17.13	0.000	.4898665	.6164599
3#50+	.3920132	.0438199	8.95	0.000	.3061277	.4778986
4#30-39	.6200306	.0420412	14.75	0.000	.5376313	.7024299
4#40-49	.4657831	.0366039	12.72	0.000	.3940407	.5375255
4#50+	.3122976	.0429351	7.27	0.000	.2281463	.396449
5#30-39	.5347281	.0580266	9.22	0.000	.4209981	.6484581
5#40-49	.3804535	.0470786	8.08	0.000	.2881812	.4727259
5#50+	.2423312	.0449042	5.40	0.000	.1543206	.3303417
6#30-39	.4473447	.0744291	6.01	0.000	.3014663	.593223
6#40-49	.3019213	.0564876	5.34	0.000	.1912078	.4126349
6#50+	.1838493	.0458435	4.01	0.000	.0939977	.2737008
7#30-39	.3630971	.0867003	4.19	0.000	.1931676	.5330267
7#40-49	.2334903	.0613363	3.81	0.000	.1132733	.3537073
7#50+	.1369302	.0443075	3.09	0.002	.050089	.2237714
8#30-39	.2864909	.092363	3.10	0.002	.1054627	.4675191
8#40-49	.1766445	.0611475	2.89	0.004	.0567977	.2964914
8#50+	.1005104	.0405792	2.48	0.013	.0209766	.1800442
9#30-39	.2204527	.0911719	2.42	0.016	.0417591	.3991463
9#40-49	.1312684	.05699	2.30	0.021	.01957	.2429668
9#50+	.0729585	.0355358	2.05	0.040	.0033095	.1426075
10#30-39	.1660933	.0845261	1.96	0.049	.0004252	.3317615
10#40-49	.0961867	.0504241	1.91	0.056	-.0026428	.1950161
10#50+	.0525181	.0300345	1.75	0.080	-.0063483	.1113846
11#30-39	.1230226	.0745199	1.65	0.099	-.0230338	.269079
11#40-49	.0697281	.0428698	1.63	0.104	-.0142953	.1537515
11#50+	.0375723	.0246919	1.52	0.128	-.010823	.0859677


```
. mlistat
at() values held constant
```

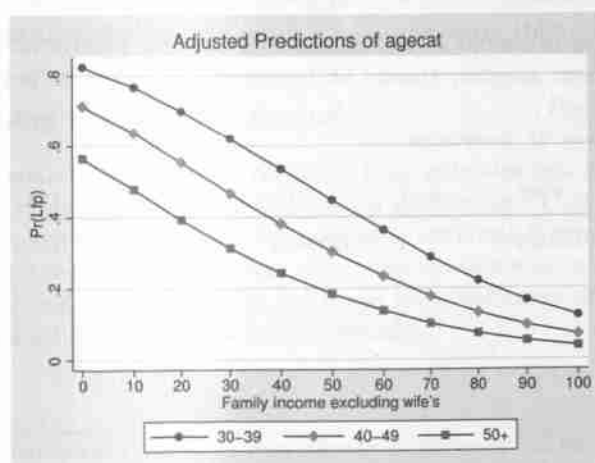
k5	k618	2. agecat	3. agecat	1. wc	1. hc	lwg
.238	1.35	.385	.219	.282	.392	1.1

```
at() values vary
```

_at	inc
1	0
2	10
3	20
4	30
5	40
6	50
7	60
8	70
9	80
10	90
11	100

The labeling of the predictions from `margins` can be confusing. The left column of the prediction table is labeled `_at#agecat`, which indicates that the information in this column begins with a number corresponding to the 11 values of `inc` used for making predictions; these are referred to as the `_at` values. For example, 1 is the prediction with `inc=0` while 11 is the prediction with `inc=100`. After the "#", the value or value label for `agecat` is listed. For example, the row labeled `1#30-39` contains the predictions when all variables except `agecat` are held at the first `_at` value, with `agecat=1` as indicated by the value label `30-39`. The command `marginsplot`, `noci` automatically understands what these predictions are and creates the plot we want:

```
. marginsplot, noci legend(cols(3))
Variables that uniquely identify margins: inc agecat
```



This example shows that `marginsplot` can plot multiple curves for the same outcome from the same model. Unfortunately, it cannot plot curves for multiple outcomes (for example, the probability of categories 1, 2, and 3 in an ordinal model) or predictions from different models (for example, showing how the predictions differ from two specifications of the model). For this, you need to use `mgen`.

Using `mgen` with `graph`

We can create the same graph as in the previous section by running `mgen` once for each level of `agecat`:

```
. mgen, atmeans at(inc=(0(10)100) agecat=1) stub(PLT1) predlab(30 to 39)
Predictions from: margins, atmeans at(inc=(0(10)100) agecat=1) predict(pr)
Variable   Obs Unique      Mean      Min      Max  Label
-----
PLT1pr1    11      11  .4591184  .1230226  .8236541  30 to 39
PLT1l1l1    11      11  .3349719  -.0230338  .7624032  95% lower limit
PLT1u1l1    11      11  .583265   .269079   .8849049  95% upper limit
PLT1inc     11      11      50         0      100  Family income excluding...
```

Specified values of covariates

	k5	k618	agecat	1. wc	1. hc	lwg
	.2377158	1.353254	1	.2815405	.3917663	1.097115

```
. mgen, atmeans at(inc=(0(10)100) agecat=2) stub(PLT2) predlab(40 to 49)
Predictions from: margins, atmeans at(inc=(0(10)100) agecat=2) predict(pr)
(output omitted)
```

Specified values of covariates

	k5	k618	agecat	1. wc	1. hc	lwg
	.2377158	1.353254	2	.2815405	.3917663	1.097115

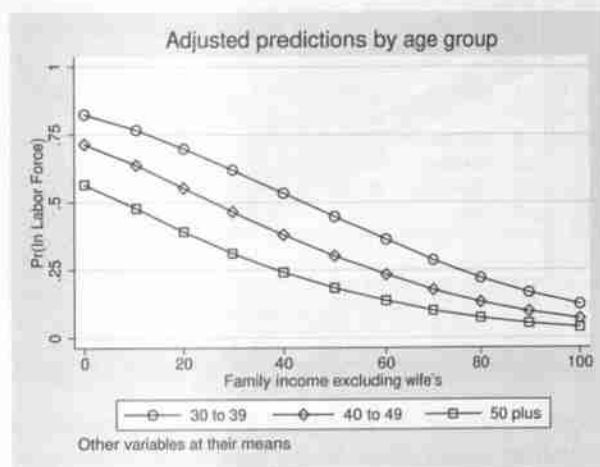
```
. mgen, atmeans at(inc=(0(10)100) agecat=3) stub(PLT3) predlab(50 plus)
Predictions from: margins, atmeans at(inc=(0(10)100) agecat=3) predict(pr)
(output omitted)
```

Specified values of covariates

	k5	k618	agecat	1. wc	1. hc	lwg
	.2377158	1.353254	3	.2815405	.3917663	1.097115

Then, we use a more complex `graph` command to obtain the following graph:

```
. graph twoway connected PLT1pr PLT2pr PLT3pr PLTinc,
> title("Adjusted predictions by age group")
> caption("Other variables at their means")
> msym(Oh Dh Sh) msiz(*1.4 *1.1 *1.2) mcol(black black black)
> lpat(solid solid solid)
> ytitle("Pr(In Labor Force)") ylab(0(.25)1, grid gmin gmax)
> legend(cols(3))
```



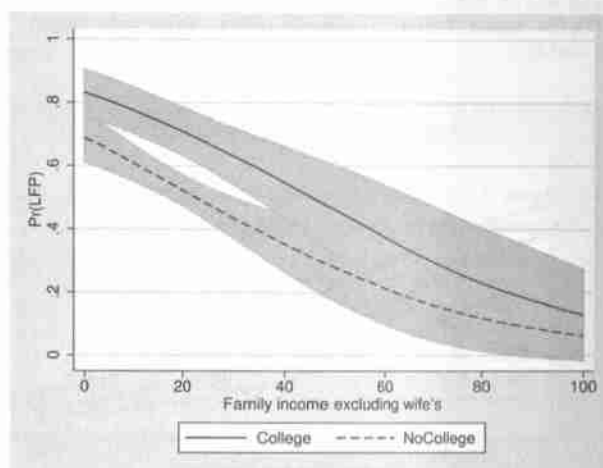
When there are multiple sets of lines and symbols to be drawn—in this case, three sets—you need to provide options for each set. The option `msym(Oh Dh Sh)` indicates that we want large, hollow circles, diamonds, and squares for the symbols. We find that `Oh` by default is a smaller symbol than `Dh` or `Sh`. The `msize()` option lets you specify the size for each symbol. Although you can use names for the sizes, such as `msize(large medium medsmall)`, we find relative sizing to be easier. Our option `msize(*1.4 *1.1 *1.2)` tells `graph` to make the first symbol 1.4 times larger than normal, and so on.

6.6.4 Overlapping confidence intervals

We find that researchers sometimes conclude that estimates are significantly different only if confidence intervals for two estimates do not overlap. That is, if the confidence intervals overlap, the hypothesis that the estimates are equal is accepted. Although this might have been a useful approximation when computation was very expensive, it often leads to incorrect conclusions because it ignores the covariances of the estimators that need to be taken into account when testing equality.⁴

4. Schenker and Gentleman (2001) show that inference is conservative if the estimators are independent.

To illustrate the problem, as well as show how discrete changes and marginal changes can be graphed, we use the techniques above to plot the probability of labor force participation by income for women who attended college and those who did not. We start with the graph before showing how we created it:



We use one `mgen` command for each level of `wc`:

```
. mgen, atmeans at(inc=(0(5)100) wc=0) stub(PLTWC0) predlab(NoCollege)
Predictions from: margins, atmeans at(inc=(0(5)100) wc=0) predict(pr)
Variable   Obs Unique   Mean   Min   Max   Label
-----
PLTWC0pr1  21     21   .3177494   .0623648   .6889161   NoCollege
PLTWC0l1l  21     21   .2309727  -.0151898   .6107004   95% lower limit
PLTWC0ul1  21     21   .404526   .1399194   .7671317   95% upper limit
PLTWC0inc  21     21     .50         0       100   Family income excluding...
```

Specified values of covariates

k5	k618	2. agecat	3. agecat	wc	1. hc	lw
.2377158	1.353254	.3851262	.2191235	0	.3917663	1.097115

```
. mgen, atmeans at(inc=(0(5)100) wc=1) stub(PLTWC1) predlab(College)
(output omitted)
```

We then combine two `rarea` graphs with a `connected` graph:

```
. twoway
> (rarea PLTWC0ul PLTWC0l1 PLTWC0inc, col(gs12))
> (rarea PLTWC1ul PLTWC1l1 PLTWC0inc, col(gs12))
> (connected PLTWC0pr PLTWC1pr PLTWC0inc, msym(i i) lpat(dash solid))
> , ytitle(Pr(In Labor Force)) legend(order(4 3))
```


Judging by the overlap of confidence intervals, we might mistakenly conclude that the probability of labor force participation was significantly higher for women who attended college when family income was between \$5,000 and \$40,000 but not at other incomes.

To see how poorly this “approximation” works, we compute the discrete change conditional on income with `mgen`. The option `dydx(wc)` specifies that we want to predict the marginal effect of `wc`. Because `wc` was entered into the model as the factor variables `i.wc`, `mgen` computes a discrete change.

```
. mgen, dydx(wc) atmeans at(inc=(0(5)100)) stub(PLTWCDG)
>      predlab(Discrete change in LFP by attending college)
Predictions from: margins, dydx(wc) atmeans at(inc=(0(5)100)) predict(pr)
Variable      Obs Unique      Mean      Min      Max      Label
-----
PLTWCDGd_pr1   21      21   .1507267   .066319   .1967745 Discrete change in L...
PLTWCDGcll1    21      21   .0556941  -.0111785   .0895455 95% lower limit
PLTWCDGcul1    21      21   .2457593   .1438166   .3049388 95% upper limit
PLTWCDGinc     21      21      50         0      100 Family income exclud...

Specified values of covariates
      k5      k618      2.      3.      1.      1.      lvg
      agecat      agecat      wc      hc
-----
.2377158  1.353254  .3851262  .2191235  .2815405  .3917663  1.097115
```

Plotting the results along with those for the probabilities leads to figure 6.2, which shows that women who attended college have significantly higher probabilities of labor force participation over almost the entire income distribution, excepting only incomes above \$95,000 (where there are very few cases). What is remarkable about `margins` is that it allows you to test just about anything you might want to say about your predictions!

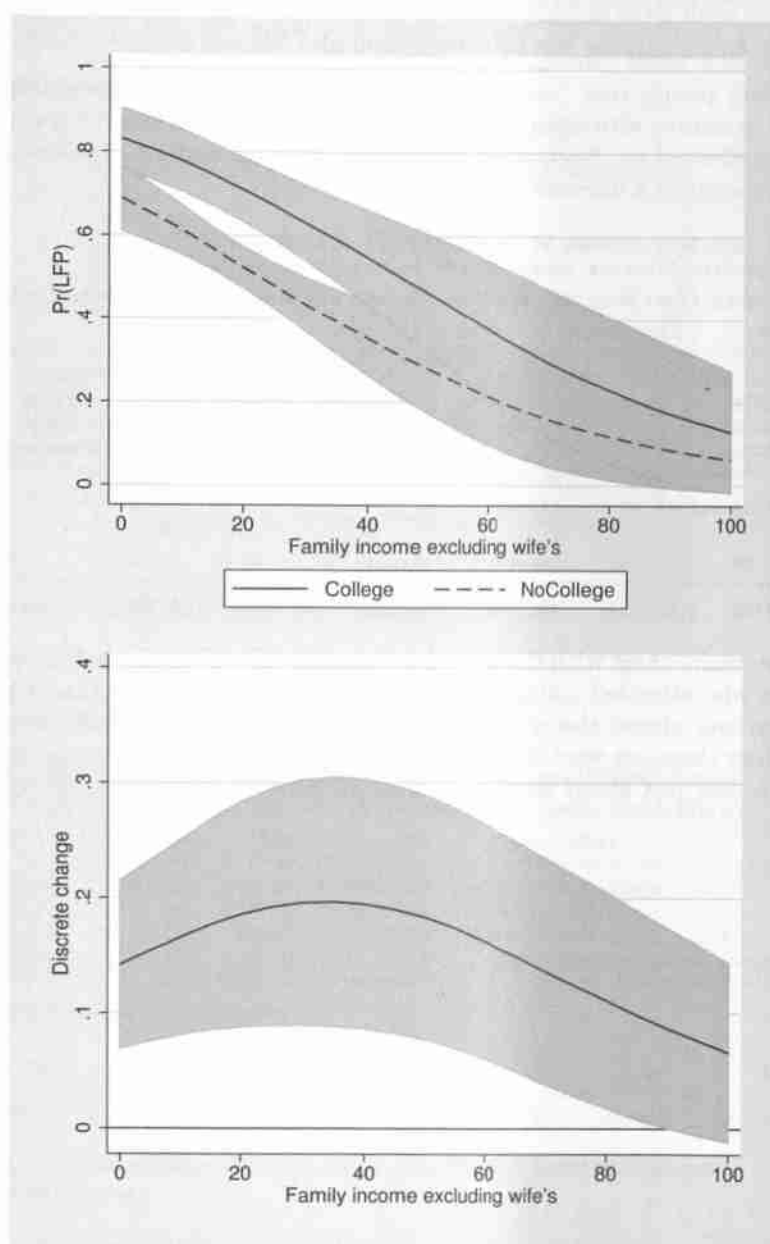


Figure 6.2. Overlapping confidence intervals compared with discrete change

6.6.5 Adding power terms and plotting predictions

As shown in section 6.2.1, squared terms can be included in models by using factor-variable notation. For example, income and income-squared can be included in the model by adding the term `c.inc##c.inc`. Although you can obtain the same parameter estimates by generating a new variable for income-squared, `margins` or our `m*` commands will not compute predictions correctly. With factor-variable notation, however, power terms and interaction terms do not pose any special problems.⁵ When `mgen` makes predictions, it automatically increases income-squared appropriately as income changes.

To illustrate how this works, we compare predictions from a model that is linear in `inc` with a model that adds the squared term `c.inc#c.inc`. First, we fit the model that includes income (but not income-squared) and make predictions:

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg inc, nolog
      (output omitted)
. mgen, predlabel(linear) atmeans at(inc=(0(10)100)) stub(_lin)
Predictions from: margins, atmeans at(inc=(0(10)100)) predict(pr)
```

Variable	Obs	Unique	Mean	Min	Max	Label
_linpri	11	11	.3608011	-.0768617	.7349035	linear
_linl11	11	11	.2708139	-.0156624	.6641427	95% lower limit
_linu11	11	11	.4507883	.1693859	.8056643	95% upper limit
_lininc	11	11	50	0	100	Family income excluding...

Specified values of covariates

k5	k618	2. agecat	3. agecat	1. wc	1. hc	lwg
.2377158	1.353254	.3851262	.2191235	.2815405	.3917663	1.097115

5. With `prgen`, our earlier command for graphing predictions, you could not generate correct predictions when your model included, for example, income and income-squared. With minor programming, our command `praccum` allowed you to generate the correct predictions. Our impression is that it was not used often.

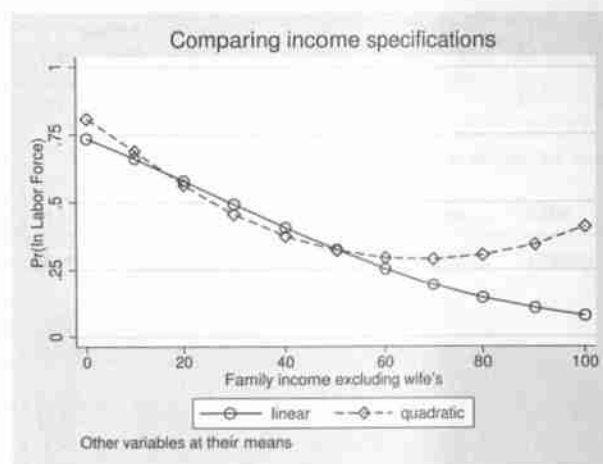
Next, we fit the model that adds income-squared and make predictions:

```
. logit lfp k5 k618 i.agecat i.wc i.hc lwg c.inc##c.inc, nolog
(output omitted)
. mgen, predlabel(quadratic) atmeans at(inc=(0(10)100)) stub(_quad)
Predictions from: margins, atmeans at(inc=(0(10)100)) predict(pr)
Variable   Obs Unique   Mean   Min   Max   Label
-----
_quadpr1   11     11   .4410442   .2887324   .8078035   quadratic
_quadll1   11     11   .2613508  -.1501808   .7207593   95% lower limit
_quadul1   11     11   .6207375   .4265932   .9690264   95% upper limit
_quadinc   11     11     50         0       100   Family income excluding...
```

Specified values of covariates

	2.	3.	1.	1.		
k5	k618	agecat	agecat	wc	hc	lwg
.2377158	1.353254	.3851262	.2191235	.2815405	.3917663	1.097115

Then, we plot the predictions:



Although the differences at higher incomes are suggestive and dramatic, the evidence for preferring the quadratic model is mixed. BIC provides positive support for the linear model, while AIC supports the quadratic model. The coefficient for income-squared is significant at the 0.046 level. The confidence intervals around the predictions at high income levels (not shown) are wide. Based on these results, we are not convinced to abandon our baseline model.

6.6.6 (Advanced) Graphs with local means

When plotting predictions over the range of a variable, you must decide where to hold the values of other variables. With the `atmeans` option in `mgen`, as the plotted variable changes, the other variables stay at the same global means. Following our previous discussion of local means, in this section we show you how to allow the values of the other variables to change as the variable being plotted changes. This requires using `mtable` with the `over()` option and moving predictions from the matrix that `mtable` returns. These steps require more data management than other parts of the book, but they can provide valuable insights into how robust your plot and conclusions are to assumptions about the levels of other variables.

When demonstrating tables of predictions, we suggested caution before holding other variables at their global means because changing one variable while holding all other variables at the same values might not be realistic. For example, suppose that we included age in our model as a continuous variable ranging from 20 to 90. Plotting predictions as age changes while holding the number of young children constant is unrealistic because older respondents are unlikely to have any young children in the family. Note that we have the same problem if we used `asobserved` instead of `atmeans` here; in that case, we would be including in our average predictions those cases for which the hypothetical value of age is implausible given the observed numbers of children. One alternative approach, which we will not explore further here, is to forgo using global means in favor of a set of representative values that are substantively plausible for all values of age (that is, a family with no children).

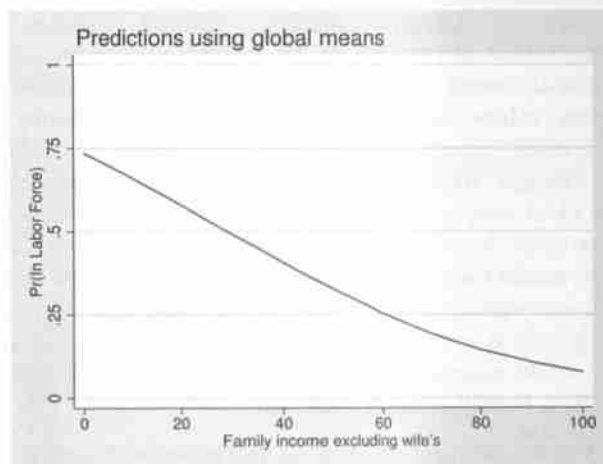
In any event, if you are plotting predictions in regions of your data where it is impossible or very unlikely that observations will exist, the predictions might be misleading. You can determine whether global means are reasonable by exploring how other values affect the results. We will consider what happens when we use local means instead of global in generating the plot. To illustrate how this is done, we start with the example used above, where we plotted labor force participation by income:


```
. mgen, at(inc=(0(10)100)) atmeans stub(GLOBAL) predlabel(Global means)
Predictions from: margins, at(inc=(0(10)100)) atmeans predict(pr)
Variable   Obs Unique   Mean      Min      Max   Label
-----
GLOBALpr1   11     11 .3608011 .0768617 .7349035 Global means
GLOBALl1l1   11     11 .2708139 -.0156624 .6641427 95% lower limit
GLOBALu1l1   11     11 .4507883 .1693859 .8056643 95% upper limit
GLOBALinc    11     11      50       0     100 Family income excluding...
```



```
Specified values of covariates
      k5      k518      agecat      agecat      1.      1.      lwg
      k5      k518      agecat      agecat      wc      hc
-----
.2377158  1.353254 .3851262 .2191235 .2815405 .3917663 1.097115
```

Plotting the predictions produces the following plot:



These predictions were made by increasing family incomes from \$0 to \$100,000, holding `wc`, `hc`, `lwg`, and other variables at their global means. This implies that those with no income have the same education and wages as those with \$100,000. As noted, before accepting this graph as a reasonable summary of the effect of family income on labor force participation, we want to determine how sensitive the predictions are to the values at which we held the other variables. In particular, what would happen if we held the other variables at levels more typical of those with a given income?

Because `inc` is continuous, we cannot compute means for the nonincome variables conditional on a single value of income, because these means might be based on very few observations. Instead, we begin by generating the variable `inc10k`, which divides `inc` into groups of \$10,000:


```
. gen inc10k = trunc(inc/10)
. label var inc10k "income in 10K categories"
. tabulate inc10k, miss
```

income in 10K categories	Freq.	Percent	Cum.
0	99	13.15	13.15
1	353	46.88	60.03
2	198	26.29	86.32
3	61	8.10	94.42
4	22	2.92	97.34
5	10	1.33	98.67
6	3	0.40	99.07
7	4	0.53	99.60
8	1	0.13	99.73
9	2	0.27	100.00
Total	753	100.00	

Because there are very few cases in some of the higher income groupings, we might want to use larger groups. But for the experiment we have in mind, this is a reasonable place to begin. Next, we use `mtable, over(inc10k) atmeans ci` to compute predictions by selecting observations at each value of `inc10k`. This is equivalent to running `mtable` repeatedly for subsamples defined by `inc10k`.

```
mtable if inc10k==0, atmeans ci
mtable if inc10k==1, atmeans ci
(output omitted)
mtable if inc10k==9, atmeans ci
```


The results show how the values of other variables vary as income changes:

```
. mtable, over(inc10k) atmeans ci
```

```
Expression: Pr(lfp), predict()
```

	k5	k618	2. agecat	3. agecat	1. wc	1. hc
1	.202	1.43	.303	.222	.121	.0808
2	.261	1.29	.363	.215	.212	.312
3	.192	1.37	.455	.192	.369	.52
4	.213	1.54	.361	.311	.443	.689
5	.318	1.55	.409	.273	.5	.727
6	.4	.6	.6	.2	.8	.8
7	0	1	.333	.667	.333	.667
8	.75	1.25	.75	0	.75	1
9	0	2	1	0	0	0
10	1	2.5	0	0	1	1

	lw	inc	Pr(y)	ll	ul
1	.922	7.25	0.641	0.584	0.698
2	1.08	15.1	0.600	0.559	0.642
3	1.17	24.1	0.588	0.546	0.630
4	1.16	33.7	0.492	0.426	0.557
5	1.05	43.4	0.373	0.283	0.463
6	1.48	53.8	0.389	0.261	0.517
7	1.07	64.9	0.201	0.085	0.318
8	1.41	75.3	0.164	0.046	0.282
9	1.07	88	0.102	-0.005	0.208
10	1.33	93.5	0.112	-0.006	0.229

Specified values where .n indicates no values specified with at()

	No at()
Current	.n

To plot these predictions, we need to move them into variables, something that is ordinarily done automatically by `mgen`. Unfortunately, `mgen` does not allow local means when making predictions. So we must manually move the predictions that `mtable` saves in the return matrix `r(table)`.

To do this, we first create a matrix, `localpred`, equal to `r(table)`. This is necessary because some of the commands we want to use will not work on an `r()` matrix.


```
. matrix localpred = r(table)
. matlist localpred, format(%8.2g)
```

	k5	k618	2. agecat	3. agecat	1. wc	1. hc
1	.2	1.4	.3	.22	.12	.081
2	.26	1.3	.36	.22	.21	.31
3	.19	1.4	.45	.19	.37	.52
4	.21	1.5	.36	.31	.44	.69
5	.32	1.5	.41	.27	.5	.73
6	.4	.6	.6	.2	.8	.8
7	0	1	.33	.67	.33	.67
8	.75	1.3	.75	0	.75	1
9	0	2	1	0	0	0
10	1	2.5	0	0	1	1

	lwg	inc	Pr(y)	11	ul
1	.92	7.2	.64	.58	.7
2	1.1	15	.6	.56	.64
3	1.2	24	.59	.55	.63
4	1.2	34	.49	.43	.56
5	1	43	.37	.28	.46
6	1.5	54	.39	.26	.52
7	1.1	65	.2	.085	.32
8	1.4	75	.16	.046	.28
9	1.1	88	.1	-.0048	.21
10	1.3	94	.11	-.0058	.23

Next, we select all rows of the matrix (designated as 1...) and the set of columns starting with column `inc` and ending with column `ul`. We use `matrix colnames` to give the columns the names we want to use for the new variables created in the next step.

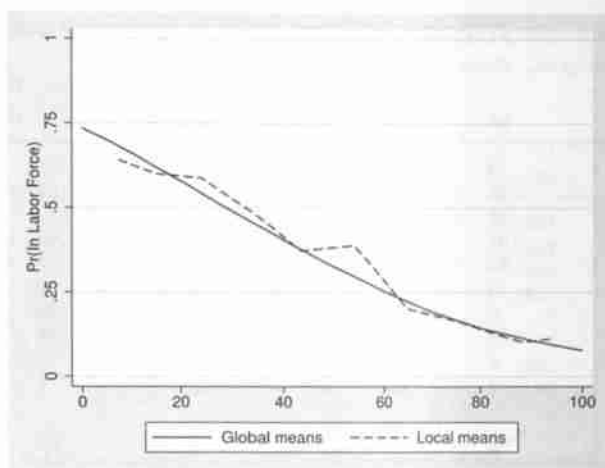
```
. matrix localpred = r(table)
. matrix localpred = localpred[1...,"inc".."ul"]
. matrix colnames localpred = LOCALinc LOCALpr LOCAL11 LOCALul
```

The command `svmat` generates variables from the columns of a matrix. The `names(col)` option specifies that the new variables should have names corresponding to the columns of the matrix.

```
. svmat localpred, names(col)
. label var LOCALpr "Local means"
```

We gave variable `LOCALpr` a label that will be used in the graph we now create:

```
. twoway
> (connected GLOBALpr GLOBALinc,
>   cicol(black) clpat(solid) msym(i))
> (connected LOCALpr LOCALinc,
>   cicol(black) clpat(dash) msym(i))
> , ytitle("Pr(In Labor Force)") ylab(0(.25)1, grid gmin gmax)
```

In this example, there are no major discrepancies, suggesting that using the global means is appropriate for making the predictions.

6.7 Conclusion

As we have discussed, the models for binary outcomes discussed in this chapter are not just widely used, they also provide a conceptual foundation for the models that we discuss in subsequent chapters. The coefficients for these models cannot be effectively interpreted directly, and even though the exponentiated coefficients for the logit model—the odds ratio—are widely used, we think these often do not convey the substance of one's findings very well either.

Instead, we spent most of the chapter describing interpretations based on predicted probabilities, and here Stata's `margins` command and our `m*` commands based on `margins` are extremely useful. `margins` is so flexible that it allows the estimation of quantities that have quite subtle conceptual differences, which may or may not have much consequence in any particular application. Regardless, though, a prerequisite on being able to clearly present findings to your audience is to understand them clearly yourself—and so understanding precisely the differences between, for example, AMEs and MEMs, is important for coherently and accurately recounting results. We also showed how to very broadly conduct hypothesis tests about different predictions that `margins` generates, and how to use `margins` with local means instead of relying simply on global means. In doing so, we hope not only to provide a strong set of tools for interpreting the results of models for binary outcomes, but also to provide a foundation for what we will do in the chapters that follow.

7 Models for ordinal outcomes

Although the categories for an ordinal variable can be ordered, the distances between the categories are unknown. For example, in survey research, questions often provide the response categories of strongly agree, agree, disagree, and strongly disagree, but an analyst would probably not assume that the distance between strongly agreeing and agreeing is the same as the distance between agreeing and disagreeing. Educational attainments can be ordered as elementary education, high school diploma, college diploma, and graduate or professional degree. Ordinal variables also commonly result from limitations of data availability that require a coarse categorization of a variable that could, in principle, have been measured on an interval scale. For example, we might have a measure of income that is simply low, medium, or high.

Ordinal variables are often coded as consecutive integers from 1 to the number of categories. Perhaps because of this coding, it is tempting to analyze ordinal outcomes with the linear regression model (LRM). However, an ordinal dependent variable violates the assumptions of the LRM, which can lead to incorrect conclusions, as demonstrated strikingly by McKelvey and Zavoina (1975, 117) and Winship and Mare (1984, 521–523). With ordinal outcomes, it is much better to use models that avoid the assumption that the distances between categories are equal. Although many models have been designed for ordinal outcomes, in this chapter we focus on the logit and probit versions of the ordinal regression model (ORM). The model was introduced by McKelvey and Zavoina (1975) in terms of an underlying latent variable, and in biostatistics by McCullagh (1980), who referred to the logit version as the proportional-odds model. In section 7.16, we review several less commonly used models for ordinal outcomes.

As with the binary regression model (BRM), the ORM is nonlinear, and the magnitude of the change in the outcome probability for a given change in one of the independent variables depends on the levels of all the independent variables. As with the BRM, the challenge is to summarize the effects of the independent variables in a way that fully reflects key substantive processes without overwhelming and distracting detail. For ordinal outcomes, as well as for the models for nominal outcomes in chapter 8, the difficulty of this task is increased by having more than two outcomes to explain.

Before proceeding, we caution that researchers should think carefully before concluding that their outcome is indeed ordinal. Do not assume that a variable should be analyzed as ordinal simply because the values of the variable can be ordered. A variable that can be ordered when considered for one purpose could be unordered or ordered differently when used for another purpose. Miller and Volker (1985) show how different assumptions about the ordering of occupations result in different conclusions.

A variable might also reflect ordering on more than one dimension, such as attitude scales that reflect both the intensity and the direction of opinion. Moreover, surveys commonly include the category “don’t know”, which probably does not correspond to the middle category in a scale, even though analysts might be tempted to treat it this way. In general, ORMs restrict the nature of the relationship between the independent variables and the probabilities of outcome categories, as discussed in section 7.15. Even when an outcome seems clearly to be ordinal, such restrictions can be unrealistic, as illustrated in chapter 8. Indeed, we suggest that you always compare the results from ordinal models with those from a model that does not assume ordinality.

We begin by reviewing the statistical model, followed by an examination of testing, fit, and methods of interpretation. These discussions are intended as a review for those who are familiar with the models. For a complete discussion, see Agresti (2010), Long (1997), or Hosmer, Lemeshow, and Sturdivant (2013). As explained in chapter 1, you can obtain sample do-files and data files by installing the `spost13.do` package.

7.1 The statistical model

The ORM can be developed in different ways, each of which leads to the same form of the model. These approaches to the model parallel those for the BRM. Indeed, the BRM can be viewed as a special case of the ordinal model in which the ordinal outcome has only two categories.

7.1.1 A latent-variable model

The ORM is commonly presented as a latent-variable model. Defining y^* as a latent variable ranging from $-\infty$ to ∞ , the structural model is

$$y_i^* = \mathbf{x}_i\beta + \varepsilon_i$$

where i is the observation and ε is a random error, as discussed further below. For the case of one independent variable,

$$y_i^* = \alpha + \beta x_i + \varepsilon_i$$

The measurement model for binary outcomes from chapter 5 is expanded to divide y^* into J ordinal categories,

$$y_i = m \quad \text{if } \tau_{m-1} \leq y_i^* < \tau_m \quad \text{for } m = 1 \text{ to } J$$

where the cutpoints τ_1 through τ_{J-1} are estimated. (Some authors refer to these as thresholds.) We assume $\tau_0 = -\infty$ and $\tau_J = \infty$ for reasons that will be clear shortly.

To illustrate the measurement model, consider the example used in this chapter. People are asked to respond to the following statement:

If you were asked to use one of four names for your social class, which would you say you belong in: the lower class, the working class, the middle class, or the upper class?

The underlying, continuous latent variable can be thought of as the propensity to identify oneself as having higher socioeconomic standing. The observed response categories are tied to the latent variable by the measurement model

$$y_i = \begin{cases} 1 \Rightarrow \text{Lower} & \text{if } \tau_0 = -\infty \leq y_i^* < \tau_1 \\ 2 \Rightarrow \text{Working} & \text{if } \tau_1 \leq y_i^* < \tau_2 \\ 3 \Rightarrow \text{Middle} & \text{if } \tau_2 \leq y_i^* < \tau_3 \\ 4 \Rightarrow \text{Upper} & \text{if } \tau_3 \leq y_i^* < \tau_4 = \infty \end{cases}$$

Thus when the latent y^* crosses a cutpoint, the observed category changes. Anderson (1984) referred to ordinal variables created in this fashion as grouped continuous variables and referred to the ORM as the grouped continuous model.

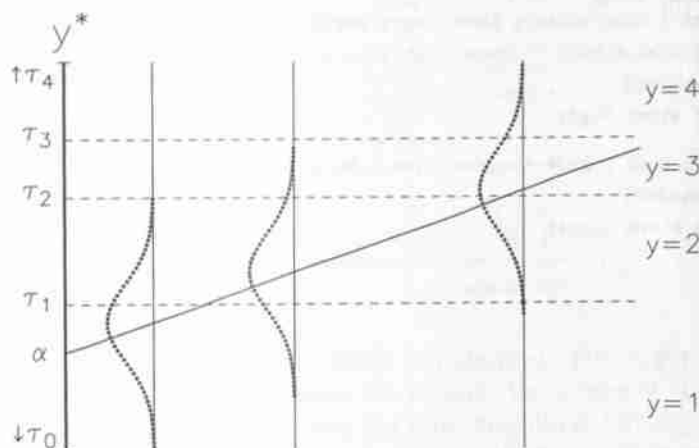


Figure 7.1. Relationship between observed y and latent y^* in ORM with one independent variable

For a single independent variable, the structural model is $y^* = \alpha + \beta x + \varepsilon$, which is plotted in figure 7.1 along with the cutpoints for the measurement model. This figure is similar to that for the BRM, except that there are now three horizontal lines representing the cutpoints τ_1 , τ_2 , and τ_3 . The three cutpoints lead to four levels of y that are labeled on the right-hand side of the graph.

The probability of an observed outcome y for a given value of x , represented by the three vertical lines in the figure, is the area under the curve between a pair of cutpoints.

For example, the probability of observing $y = m$ for given values of the x 's corresponds to the region of the distribution where y^* falls between τ_{m-1} and τ_m :

$$\Pr(y = m \mid \mathbf{x}) = \Pr(\tau_{m-1} \leq y^* < \tau_m \mid \mathbf{x})$$

Substituting $\mathbf{x}\beta + \varepsilon$ for y^* and using some algebra leads to the standard formula for the predicted probability in the ORM,

$$\Pr(y = m \mid \mathbf{x}) = F(\tau_m - \mathbf{x}\beta) - F(\tau_{m-1} - \mathbf{x}\beta) \quad (7.1)$$

where F is the cumulative distribution function for ε . In the ordinal probit model, F is normal with $\text{Var}(\varepsilon) = 1$; in the ordinal logit model, F is logistic with $\text{Var}(\varepsilon) = \pi^2/3$. For $y = 1$, the second term on the right drops out because $F(-\infty - \mathbf{x}\beta) = 0$, and for $y = J$, the first term equals $F(\infty - \mathbf{x}\beta) = 1$.

Comparing these equations with those for the BRM shows that the ORM is identical to the BRM with one exception. To demonstrate this, we fit chapter 5's binary model for labor force participation with both logit and ologit (the command for ordinal logit):

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. logit lfp c.k5 c.k618 i.agecat i.wc i.hc c.lwg c.inc, nolog
(output omitted)
. estimates store logit

. ologit lfp c.k5 c.k618 i.agecat i.wc i.hc c.lwg c.inc, nolog
(output omitted)
. estimates store ologit
```


To compare the coefficients, we use `estimates table`¹:

```
. estimates table logit ologit, b(%9.3f) t varlabel varwidth(30) equations(1:1)
```

	Variable	logit	ologit
#1	# kids < 6	-1.392	-1.392
		-7.25	-7.25
	# kids 6-18	-0.066	-0.066
		-0.96	-0.96
	agecat		
	40-49	-0.627	-0.627
		-3.00	-3.00
	50+	-1.279	-1.279
		-4.92	-4.92
	wc		
	college	0.798	0.798
		3.48	3.48
	hc		
	college	0.136	0.136
		0.66	0.66
	Log of wife's estimated wages	0.610	0.610
		4.04	4.04
	Family income excluding wife's	-0.035	-0.035
		-4.24	-4.24
	Constant	1.014	
		3.54	
cut1			
	Constant		-1.014
			-3.54

legend: b/t

The slope coefficients and their z -values are identical. For `logit`, though, an intercept or constant is reported, whereas for `ologit`, the intercept is replaced by the cutpoint labeled `cut1`. The cutpoint has the same magnitude but opposite sign as the intercept from `logit`. This difference is due to how the two models are identified. As the ORM has been presented, there are “too many” free parameters; that is, you cannot estimate $J - 1$ thresholds and the constant too. For a unique set of maximum likelihood estimates to exist, an identifying assumption needs to be made about either the intercept or one of the cutpoints. Stata's `ologit` and `oprobit` commands identify the ORM by assuming that the intercept is 0 and then estimating all cutpoints.

Some other software packages that fit the ORM instead fix one of the cutpoints to 0 and estimate the intercept. Although the different parameterizations can be confusing, keep in mind that the slope coefficients and predicted probabilities are the same under

1. Because `logit` has a constant and `ologit` has a cutpoint, by default `estimates table` will not line up the coefficients from the two models. Rather, each of the independent variables will be listed twice. `equations(1:1)` tells `estimates table` to line up the coefficients. This is easiest to understand if you try our command without the `equations(1:1)` option.

either parameterization (see Long [1997, 122–123] for details). Section 7.5 shows how to fit the ORM using alternative parameterizations.

7.1.2 A nonlinear probability model

The ORM can also be developed as a nonlinear probability model without appealing to an underlying latent continuous variable. Here we show how this is done for the ordinal logit model. First, we define the odds that an outcome is less than or equal to m versus greater than m given \mathbf{x} :

$$\Omega_{\leq m | > m}(\mathbf{x}) \equiv \frac{\Pr(y \leq m | \mathbf{x})}{\Pr(y > m | \mathbf{x})} \quad \text{for } m = 1, J - 1$$

For example, we could compute the odds of lower- or working-class identification (that is, $m \leq 2$) versus middle- or upper-class identification ($m > 2$). The log of the odds is assumed to equal

$$\ln \Omega_{\leq m | > m}(\mathbf{x}) = \tau_m - \mathbf{x}\beta \quad (7.2)$$

Critically, the β 's are the same for all values of m .

For a single independent variable and three categories, where we are fixing the intercept to equal 0 and estimating the τ 's, the model is

$$\begin{aligned} \ln \frac{\Pr(y \leq 1 | \mathbf{x})}{\Pr(y > 1 | \mathbf{x})} &= \tau_1 - \beta x \\ \ln \frac{\Pr(y \leq 2 | \mathbf{x})}{\Pr(y > 2 | \mathbf{x})} &= \tau_2 - \beta x \end{aligned}$$

Although it may seem confusing that we subtract βx rather than adding it, this is a consequence of computing the logit of $y \leq m$ versus $y > m$. We agree that it would be simpler to stick with $\tau_m + \beta x$, but this is not the way the model is normally presented.

7.2 Estimation using ologit and oprobit

The ordered logit and probit models can be fit with the following commands and their basic options:

```
ologit depvar [indepvars] [if] [in] [weight] [, vce(vcetype) or]
```

```
oprobit depvar [indepvars] [if] [in] [weight] [, vce(vcetype)]
```

In our experience, these models take more steps to converge than either models for binary outcomes fit using `logit` or `probit` or models for nominal outcomes fit using `mlogit`.

Variable lists

depvar is the dependent variable. The values assigned to the outcome categories are irrelevant, except that larger values are assumed to correspond to “higher” outcomes. For example, if you had three outcomes, you could use the values 1, 2, and 3, or -1.23, 2.3, and 999. To avoid confusion, however, we recommend coding your dependent variable as consecutive integers beginning with 1.

indepvars is a list of independent variables. If *indepvars* is not included, Stata fits a model with only cutpoints.

Specifying the estimation sample

if and in qualifiers. These can be used to restrict the estimation sample. For example, if you want to fit an ordered logit model for only those surveyed in 1980 (*year* = 1), you could specify `ologit class i.female i.white i.educ age inc if year==1`.

Listwise deletion. Stata excludes cases in which there are missing values for any of the variables in the model. Accordingly, if two models are fit using the same dataset but have different sets of independent variables, it is possible to have different samples. We recommend that you use `mark` and `markout` (discussed in section 3.1.6) to explicitly remove cases with missing data.

Weights and complex samples

Both `ologit` and `oprobit` can be used with `fweights`, `pweights`, and `iweights`. Survey estimation can be done using the `svy` prefix. See section 3.1.7 for details.

Options

`vcetype` specifies the type of standard errors to be computed. See section 3.1.9 for details.

or reports odds ratios for the ordered logit model.

Additional options and information can be found in the Stata manual entries [R] `ologit` and [R] `oprobit`.

7.2.1 Example of ordinal logit model

Our example is based on a question asked in the 1980, 1996, and 2012 General Social Surveys. These are repeated cross-sectional data, not panel data. That is, in each wave the survey was administered to new respondents from a new nationally representative sample. The following variables are in the model:


```
. use gssclass4, clear
(gssclass4.dta | GSS Subjective Class Identification | 2013-11-20)
. codebook class female white year ed age income, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
class	5620	4	2.437544	1	4	subjective class id
female	5620	2	.5491103	0	1	respondent is female
white	5620	2	.8140569	0	1	respondent is white
year	5620	3	2.070996	1	3	year of GSS survey
educ	5620	3	2.064769	1	3	educational attainment
age	5620	72	45.15712	18	89	age of respondent
income	5620	62	68.07737	.51205	324.2425	household income

Respondents were asked to indicate the social class to which they think they most belong, using categories coded 1 = lower, 2 = working, 3 = middle, and 4 = upper. The resulting variable `class` has the distribution:

```
. tabulate class
```

subjective class id	Freq.	Percent	Cum.
lower	394	7.01	7.01
working	2,567	45.68	52.69
middle	2,465	43.86	96.55
upper	194	3.45	100.00
Total	5,620	100.00	

The variable `educ` is a categorical variable in which the categories are less than a high school diploma, high school diploma, and college diploma. The variable `income` is measured in 2012 dollars for all years of the survey.

Using these data, we use `ologit` to fit the model

$$\Pr(\text{class} = m \mid \mathbf{x}_i) = F(\tau_m - \mathbf{x}\beta) - F(\tau_{m-1} - \mathbf{x}\beta)$$

where

$$\begin{aligned} \mathbf{x}\beta = & \beta_{\text{female}} \text{female} + \beta_{\text{white}} \text{white} \\ & + \beta_{\text{year}[1996]} (\text{year}==1996) + \beta_{\text{year}[2012]} (\text{year}==2012) \\ & + \beta_{\text{educ}[hs \text{ only}]} (\text{educ}==2) + \beta_{\text{educ}[college]} (\text{educ}==3) \\ & + \beta_{\text{c.age}} \text{c.age} + \beta_{\text{c.age}\#\text{c.age}} \text{c.age}\#\text{c.age} + \beta_{\text{income}} \text{income} \end{aligned}$$

To specify the model, we use the factor-variable notation `i.year` to create indicators for the year of the survey, `i.educ` for education, and `c.age##c.age` to include age and age-squared. `estimates store` is used so that we can later make a table containing these results.


```
. ologit class i.female i.white i.year i.educ c.age##c.age income, nolog
Ordered logistic regression      Number of obs   =      5620
                                LR chi2(9)          =     1453.95
                                Prob > chi2          =      0.0000
                                Pseudo R2           =      0.1266
Log likelihood = -5016.2107
```

class	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
female						
female	.0162383	.054419	0.30	0.765	-.0904211	.1228976
white						
white	.2363442	.0721307	3.28	0.001	.0949707	.3777177
year						
1996	-.0799464	.0690368	-1.16	0.247	-.215256	.0553632
2012	-.5038717	.0764131	-6.59	0.000	-.6536386	-.3541048
educ						
hs only	.3704854	.0783189	4.73	0.000	.2169832	.5239876
college	1.565553	.0978863	15.99	0.000	1.373699	1.757406
age	-.0488039	.009194	-5.31	0.000	-.0668239	-.0307839
c.age#c.age	.0007093	.0000928	7.65	0.000	.0005275	.0008911
income	.0116206	.0005234	22.20	0.000	.0105948	.0126464
/cut1	-2.140323	.2279743			-2.587144	-1.693501
/cut2	.9152707	.2251007			.4740813	1.35646
/cut3	4.934708	.243606			4.457249	5.412167

```
. estimates store ologit
```

The information in the header and the table of coefficients is in the same form as discussed in chapters 3 and 5, with the addition of estimates for the cutpoints at the end.

Next, we fit the ordered probit model:

```
. oprobit class i.female i.white i.year i.educ c.age##c.age income, nolog
(output omitted)
. estimates store oprobit
```


Because we stored the results for both models, we can compare the results with the command `estimates table`:

```
. estimates table ologit oprobit, b(%9.3f) t varlabel varwidth(30)
```

	Variable	ologit	oprobit
class			
	female		
	female	0.016	0.002
		0.30	0.05
	white		
	white	0.236	0.106
		3.28	2.65
	year		
	1996	-0.080	-0.062
		-1.16	-1.57
	2012	-0.504	-0.302
		-6.59	-6.98
	educ		
	hs only	0.370	0.195
		4.73	4.49
	college	1.566	0.852
		15.99	15.74
	age of respondent	-0.049	-0.025
		-5.31	-4.91
	c.age#c.age	0.001	0.000
		7.65	7.03
	household income	0.012	0.006
		22.20	22.90
cut1	Constant	-2.140	-1.285
		-9.39	-10.05
cut2	Constant	0.915	0.471
		4.07	3.72
cut3	Constant	4.935	2.599
		20.26	19.58

legend: b/t

As with the BRM, the estimated coefficients differ from logit to probit by a factor of about 1.7, reflecting the different scaling of the ordered logit and ordered probit models that results from different assumptions about the variance of the errors. We also see scaling differences in the cutpoints, which are also larger in the ordered logit model. Values of the z tests are similar because they are not affected by the scaling, but they are not identical because of slight differences in the shape of the assumed distribution of the errors.

7.2.2 Predicting perfectly

If either the highest or the lowest category of the dependent variable does not vary within one of the categories of an independent variable, there will be a problem with estimation. To see what happens, we created an artificial example with a dummy variable for whether respondents have a college degree. Tabulating `college` against `class` shows that in all cases where `college` is 1, respondents have values of `class` equal to 4, indicating upper-class identification:

```
. tabulate class college
```

subjective class id	Has college degree?		Total
	no	yes	
lower	394	0	394
working	2,567	0	2,567
middle	2,465	0	2,465
upper	81	113	194
Total	5,507	113	5,620

Accordingly, if you know `college` is 1, you can predict perfectly that `class` is 4. Although we purposely constructed `college` so this would happen, perfect prediction occurs in real data, especially when samples are small or one of the outcome values is infrequent.

When we fit the ordered logit model with `college` as a regressor, the perfectly predicted observations are retained in the estimation sample with a warning message appearing below the table of estimates:


```
. ologit class i.female i.white i.year i.college c.age#c.age income, nolog
Ordered logistic regression
Log likelihood = -4897.1452
Number of obs   = 5620
LR chi2(8)      = 1692.08
Prob > chi2     = 0.0000
Pseudo R2      = 0.1473
```

class	Coef.	Std. Err.	z	P> z	[95% Conf. interval]	
female						
female	.0533943	.0545116	0.98	0.327	-.0534465	.160235
white						
white	.2431002	.0710321	3.42	0.001	.1038797	.3823206
year						
1996	.0313275	.0680653	0.46	0.645	-.1020781	.1647331
2012	-.28093	.0746029	-3.77	0.000	-.427149	-.134711
college						
yes	35.48775	731118.3	0.00	1.000	-1432930	1433001
age	-.0396543	.0092189	-4.30	0.000	-.057723	-.0215857
c.age#c.age	.0005788	.0000929	6.23	0.000	.0003966	.0007609
income	.0133525	.0005602	23.84	0.000	.0122546	.0144505
/cut1	-2.217543	.2237908			-2.656165	-1.778921
/cut2	.7238815	.2204395			.2918281	1.155935
/cut3	5.342231	.2540771			4.844249	5.840213

Note: 113 observations completely determined. Standard errors questionable.

The note reflects that the standard error for `college` is enormous, indicating the problem that occurs when trying to estimate a coefficient that is effectively infinite. Another way of thinking about the large standard error is that the lack of variation in the outcome when `college` equals 1 means we do not have any information that would permit us to estimate the coefficient with precision. When this happens, our next step is to drop the 113 cases for which `college` equals 1 (you could use the command `drop if college==1` to do this) and refit the model without `college`. This is done automatically for binary models fit by `logit` and `probit` (see section 5.2.3).

There is no problem if an independent variable perfectly predicts one of the middle categories. For example, if all observations for which `college` is 1 reported being middle class, this would not cause problems for estimation.

7.3 Hypothesis testing

Hypothesis tests of regression coefficients can be evaluated with the z statistics in the estimation output, with `test` and `testparm` for Wald tests of simple and complex hypotheses, and with `lrtest` for likelihood-ratio tests. We briefly review each. See section 3.2 for additional information on these commands.

7.3.1 Testing individual coefficients

If the assumptions of the model hold, the maximum likelihood estimators from `ologit` and `oprobit` are distributed asymptotically normally. The hypothesis $H_0: \beta_k = \beta^*$ can be tested with $z = (\hat{\beta}_k - \beta^*) / \widehat{\text{se}}_{\hat{\beta}_k}$. Under the assumptions justifying maximum likelihood, if H_0 is true, then z is distributed approximately normally with a mean of 0 and a variance of 1 for large samples. For example, consider the results for the variable `white` from the `ologit` output above. We are using the `sformat` option to show more decimal places for the z statistic:²

```
. ologit class i.female i.white i.year i.educ c.age##c.age income,
> nolog sformat(%8.3f)

Ordered logistic regression               Number of obs   =       5620
                                          LR chi2(9)       =      1453.95
                                          Prob > chi2      =       0.0000
                                          Pseudo R2       =       0.1266

Log likelihood = -5016.2107
```

class	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
female					
female	.0162383	.054419	0.298	0.765	-.0904211 .1228976
white					
white	.2363442	.0721307	3.277	0.001	.0949707 .3777177

(output omitted)

We conclude the following:

Whites and nonwhites significantly differ in their subjective social class identification ($z = 3.28$, $p < 0.01$, two-tailed).

Either a one-tailed or a two-tailed test can be used, as discussed in chapter 5.

The z test in the output of estimation commands is a Wald test, which can also be computed using `test`. For example, to test $H_0: \beta_{\text{white}} = 0$, type

```
. test 1.white
( 1) [class]1.white = 0
      chi2( 1) =    10.74
      Prob > chi2 =    0.0011
```

We conclude the following:

Whites and nonwhites significantly differ in their class identification ($\chi^2 = 10.74$, $df = 1$, $p < 0.01$).

2. We are displaying more decimal places to later demonstrate the equivalence of the z test and the corresponding chi-squared test. With any estimation command, the option `cformat(fmt)` can be used to format the display of coefficients and standard errors. Likewise, option `pformat(fmt)` formats the display of p -values and option `sformat(fmt)` formats the display of test statistics. Alternatively, the `set` command can also be used to change these formats either permanently or for the rest of the current Stata session. See [R] `set cformat` in the Stata manuals for details.

The value of a chi-squared test with 1 degree of freedom is identical to the square of the corresponding z test, which can be demonstrated with the `display` command:

```
. display "z*z=" 3.277*3.277
z*z=10.738729
```

A likelihood-ratio LR test is computed by comparing the log likelihood from a full model with that from a restricted model. To test a single coefficient, we begin by fitting the full model and storing the estimates:

```
. ologit class i.female i.white i.year i.educ c.age##c.age income, nolog
(output omitted)
. estimates store fullmodel
```

Then, we fit a model that excludes the variable `white` that we want to test.

```
. ologit class i.female i.year i.educ c.age##c.age income, nolog
(output omitted)
. estimates store dropwhite
```

The `lrtest` command computes the test:

```
. lrtest fullmodel dropwhite
Likelihood-ratio test
(Assumption: dropwhite nested in fullmodel)
LR chi2(1) = 10.75
Prob > chi2 = 0.0010
```

The resulting LR test can be interpreted as follows:

The effect of being white on class identification is significant ($LR \chi^2 = 10.75$, $df = 1$, $p < 0.01$).

7.3.2 Testing multiple coefficients

We can also test complex hypotheses that involve more than one coefficient. For example, our model has the demographic variables `white`, `female`, and `age`. To test that the effects of these variables are simultaneously equal to 0—that is, $H_0: \beta_{\text{white}} = \beta_{\text{female}} = \beta_{\text{age}} = \beta_{\text{age}\#age} = 0$ —we can use either a Wald or an LR test. For the Wald test, we fit the full model and then use the `test` command:

```
. ologit class i.female i.white i.year i.educ c.age##c.age income, nolog
(output omitted)
. test 1.white 1.female age age#age
(1) [class]1.white = 0
(2) [class]1.female = 0
(3) [class]age = 0
(4) [class]c.age#c.age = 0
      chi2( 4) = 226.75
      Prob > chi2 = 0.0000
```


Before we interpret the results of the test, we want to clarify how coefficients are specified in the `test` command when factor-variable notation is used. The specification `i.white` added the variable `i.white` to the model as shown in the output to `ologit` above. Accordingly, we are testing the coefficient associated with the variable `i.white`, not `i.white` or `white`. The same rule applies for `female`. Age was entered into the model as `c.age##c.age`, which was expanded to estimate coefficients for `c.age` and `c.age#c.age`. When entering these coefficients into `test`, we do not need to include the `c.` prefix (although we could do so). Regardless of how we specify the `test` command, we conclude the following:

The hypothesis that the demographic effects of age, race, and sex are simultaneously equal to 0 can be rejected at the 0.01 level ($\chi^2 = 226.8$, $df = 4$, $p < 0.01$).

To compute an LR test of multiple coefficients, we first fit the full model and store the results with `estimates store`. Suppose we are interested in whether demographic characteristics matter at all for subjective class identification or whether identification is only a function of socioeconomic status and changes over time. To test $H_0: \beta_{\text{white}} = \beta_{\text{female}} = \beta_{\text{age}} = \beta_{\text{age\#age}} = 0$, we fit the model that excludes these four coefficients and run `lrtest`:

```
. ologit class i.female i.white i.year i.educ c.age##c.age income, nolog
      (output omitted)
. estimates store fullmodel
. ologit class i.year i.educ income, nolog
      (output omitted)
. estimates store dropdemog
. lrtest fullmodel dropdemog

Likelihood-ratio test          LR chi2(4) =    236.53
(Assumption: dropdemog nested in fullmodel)  Prob > chi2 =    0.0000
```

We conclude the following:

The hypothesis that the demographic effects of age, race, and sex are simultaneously equal to 0 can be rejected at the 0.01 level (LR $\chi^2 = 236.5$, $df = 4$, $p < 0.01$).

We find that the Wald and LR tests usually lead to the same decisions, and there is no reason why you would typically want to compute both tests. When there are differences, they generally occur when the tests are near the cutoff for statistical significance. Because the LR test is invariant to reparameterization, we prefer the LR test when both are available. However, only the Wald test can be used if robust standard errors, probability weights, or survey estimation are used.

When a factor variable has more than two categories, such as `year` and `educ` in our model, you can specify each of the coefficients with `test` (for example, `test 2.educ 3.educ`) or you can use `testparm`:


```
. testparm i.educ
      ( 1)  [class]2.educ = 0
      ( 2)  [class]3.educ = 0
             chi2( 2) = 329.48
             Prob > chi2 = 0.0000
```

test can also be used to test the equality of coefficients, as shown in section 5.3.2.

7.4 Measures of fit using fitstat

As we discussed in greater detail in chapter 3, scalar measures of fit can be used when comparing competing models (also see Long [1997, 85–113]). Several measures can be computed after either `ologit` or `oprobit` by using the `SPost` command `fitstat`. In this example, we compare a model for class identification that includes age but not age-squared with the model we have been using that includes age-squared:

```
. ologit class i.female i.white i.year i.educ age income, nolog
      (output omitted)
. quietly fitstat, save
. ologit class i.female i.white i.year i.educ c.age##c.age income, nolog
      (output omitted)
. fitstat, diff
```

	Current	Saved	Difference
Log-likelihood			
Model	-5016.211	-5045.903	29.692
Intercept-only	-5743.186	-5743.186	0.000
Chi-square			
D (df=5608/5609/-1)	10032.421	10091.806	-59.385
LR (df=9/8/1)	1453.951	1394.566	59.385
p-value	0.000	0.000	0.000
R2			
McFadden	0.127	0.121	0.005
McFadden (adjusted)	0.124	0.119	0.005
McKelvey & Zavoina	0.284	0.274	0.011
Cox-Snell/ML	0.228	0.220	0.008
Cragg-Uhler/Nagelkerke	0.262	0.252	0.009
Count	0.605	0.600	0.005
Count (adjusted)	0.273	0.264	0.009
IC			
AIC	10056.421	10113.806	-57.385
AIC divided by N	1.789	1.800	-0.010
BIC (df=12/11/1)	10136.030	10186.781	-50.751
Variance of			
e	3.290	3.290	0.000
y-star	4.596	4.528	0.067

Note: Likelihood-ratio test assumes saved model nested in current model.

Difference of 50.751 in BIC provides very strong support for current model.

The Bayesian information criterion (BIC), Akaike's information criterion (AIC), and the LR test each provide evidence supporting the inclusion of age-squared in the model.

Using simulations, Hagle and Mitchell (1992) and Windmeijer (1995) found that with ordinal outcomes, McKelvey and Zavoina's pseudo- R^2 most closely approximates the R^2 obtained by fitting the LRM on the underlying latent variable. If you are using y^* -standardized coefficients to interpret the ORM (see section 7.8.1), McKelvey and Zavoina's R^2 could be used as a counterpart to the R^2 from the LRM.

7.5 (Advanced) Converting to a different parameterization

We mark this section as advanced because the conversion we show is likely only pertinent to readers who also work with other statistics packages that fit the model by using the alternative parameterization. The section may still be useful to strengthen your understanding of how the intercept and thresholds of these models are related, as well as how the `lincom` command works.

Earlier, we noted that the model can be identified by fixing either the intercept or one of the thresholds to equal 0. Stata sets $\beta_0 = 0$ and estimates τ_1 , whereas some programs fix $\tau_1 = 0$ and estimate β_0 . Although all quantities of interest for interpretation (for example, predicted probabilities) are the same under both parameterizations, it is useful to see how Stata can fit the model with either parameterization. We can understand how this is done with the following equation, where we are simply adding $0 = \delta - \delta$ and rearranging terms:

$$\begin{aligned}\Pr(y = m | \mathbf{x}) &= F\{\tau_m - \beta_0 - \mathbf{x}\beta + (\delta - \delta)\} - F\{\tau_{m-1} - \beta_0 - \mathbf{x}\beta + (\delta - \delta)\} \\ &= F\{(\tau_m - \delta) - (\beta_0 - \delta) - \mathbf{x}\beta\} - F\{(\tau_{m-1} - \delta) - (\beta_0 - \delta) - \mathbf{x}\beta\}\end{aligned}$$

Without further constraints, it is possible to estimate the differences $\tau_m - \delta$ and $\beta_0 - \delta$ but not the parameters τ_m and β_0 . To identify the model, Stata assumes $\delta = \beta_0$, which forces the estimate of β_0 to be 0. Some programs assume $\delta = \tau_1$, which forces the estimate of τ_1 to be 0. The following table shows the differences in the parameterizations:

Model parameter	Stata's parameterization	Alternative parameterization
β_0	$\beta_0 - \beta_0 = 0$	$\beta_0 - \tau_1$
τ_1	$\tau_1 - \beta_0$	$\tau_1 - \tau_1 = 0$
τ_2	$\tau_2 - \beta_0$	$\tau_2 - \tau_1$
τ_3	$\tau_3 - \beta_0$	$\tau_3 - \tau_1$

Although you would only need to estimate the alternative parameterization if you wanted to compare your results with those produced by another statistics package,

seeing how this is done illustrates why the intercept and thresholds are arbitrary. To estimate the alternative parameterization, we use `lincom` to compute the difference between Stata's estimates and the estimated value of the first cutpoint. We begin with the alternative parameterization of the intercept:

```
. ologit class i.female i.white i.year i.educ c.age##c.age income, nolog
(output omitted)
. lincom 0 - _b[/cut1] // intercept
(1) - [cut1]_cons = 0
```

class	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
(1)	2.140323	.2279743	9.39	0.000	1.693501	2.587144

To understand the `lincom` command, you need to know that `_b[/cut1]` is how Stata refers to the estimate of the first cutpoint. Accordingly, `0 - _b[/cut1]` is the difference between 0 and the estimate of the first cutpoint, which simply reverses the sign of the first estimated cutpoint.

For the other cutpoints, we are estimating $\tau_2 - \tau_1$ and $\tau_3 - \tau_1$, which correspond to `_b[/cut2] - _b[/cut1]` and `_b[/cut3] - _b[/cut1]`:

```
. lincom _b[/cut2] - _b[/cut1] // cutpoint 2
(1) - [cut1]_cons + [cut2]_cons = 0
```

class	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
(1)	3.055594	.0573347	53.29	0.000	2.94322	3.167968

```
. lincom _b[/cut3] - _b[/cut1] // cutpoint 3
(1) - [cut1]_cons + [cut3]_cons = 0
```

class	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
(1)	7.075031	.1097937	64.44	0.000	6.859839	7.290223

The estimate of $\tau_1 - \tau_1$ is, of course, 0. These estimates would match those from a program using the alternative parameterization.

7.6 The parallel regression assumption

Before discussing interpretation, it is important to understand an assumption that is implicit in the ORM, known both as the parallel regression assumption and, for the ordinal logit model, the proportional-odds assumption. Using (7.1), the ORM with J outcome categories can be written as

$$\begin{aligned}\Pr(y = 1 \mid \mathbf{x}) &= F(\tau_1 - \mathbf{x}\beta) \\ \Pr(y = m \mid \mathbf{x}) &= F(\tau_m - \mathbf{x}\beta) - F(\tau_{m-1} - \mathbf{x}\beta) \text{ for } m = 2 \text{ to } J-1 \\ \Pr(y = J \mid \mathbf{x}) &= 1 - F(\tau_{J-1} - \mathbf{x}\beta)\end{aligned}$$

Using these equations, the cumulative probabilities have the simple form

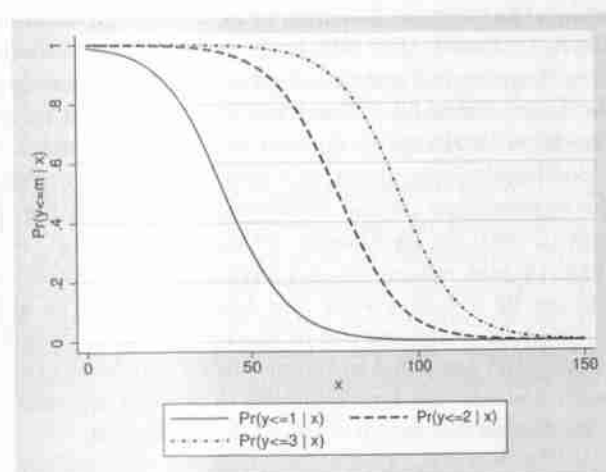
$$\Pr(y \leq m \mid \mathbf{x}) = F(\tau_m - \mathbf{x}\beta) \text{ for } m = 1 \text{ to } J-1 \quad (7.3)$$

Notice that β does not have a subscript m . Accordingly, this equation shows that the ORM is equivalent to $J-1$ binary regressions with the critical assumption that the slope coefficients are identical in each binary regression.

For example, with four outcomes and one independent variable, the cumulative probability equations are

$$\begin{aligned}\Pr(y \leq 1 \mid \mathbf{x}) &= F(\tau_1 - \beta x) \\ \Pr(y \leq 2 \mid \mathbf{x}) &= F(\tau_2 - \beta x) \\ \Pr(y \leq 3 \mid \mathbf{x}) &= F(\tau_3 - \beta x)\end{aligned}$$

Recall that the intercept α is not in the equation because it was assumed to equal 0 to identify the model. These equations lead to the following figure:



Each probability curve differs only in being shifted to the left or right. The curves are parallel as a consequence of the assumption that the β 's are equal for each equation.

This figure suggests that the parallel regression assumption can be tested by comparing the estimates from $J-1$ binary regressions,

$$\Pr(y \leq m \mid \mathbf{x}) = F(\tau_m - \mathbf{x}\beta_m) \text{ for } m = 1 \text{ to } J-1 \quad (7.4)$$

where the β 's are allowed to differ across the equations. The model in (7.4), called the generalized ORM, is discussed further in section 7.16.2. The parallel regression assumption implies that $\beta_1 = \beta_2 = \dots = \beta_{J-1}$. To the degree that the parallel regression assumption holds, the estimates $\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_{J-1}$ should be close.

There are several ways of testing the parallel regression assumption, although none of them can be done using commands in official Stata. Instead, the user-written command `oparallel` (Buis 2013), `gologit2` (Williams 2005), or `brant` (part of our `SPost` package) is required. To install any of these, while in Stata and connected to the Internet, type `net search command-name` and follow the instructions for installation.³ We begin by briefly describing the tests, and then we show how to compute them in Stata.

Under maximum likelihood theory, there are three types of tests: Wald tests, LR tests, and score tests (also called Lagrange multiplier tests). To understand how these tests are used to test the parallel regression assumption, let the generalized ORM in (7.4) be the unconstrained model and the ORM in (7.3) be the constrained model. We want to test the hypothesis $H_0: \beta_1 = \beta_2 = \dots = \beta_{J-1}$. That is, we want to test the restrictions on the unconstrained model that lead to the constrained model. A Wald test estimates the unconstrained model and tests the restrictions in the null hypothesis. The LR test estimates both the unconstrained and the constrained models and examines the change of the log likelihood. The score test estimates the constrained model and (oversimplifying some) estimates how much the log likelihood would change if the constraints were relaxed.

The command `oparallel` computes each type of test for the ordered logit model but not for the ordered probit model. The Wald test is computed by fitting a generalized ordered logit model with `gologit2` and then testing the constraints implied by parallel regressions with the `test` command. The LR test is computed by fitting a generalized ordered logit model with `gologit2` and the ordered logit model with `ologit` and comparing the log likelihoods. `oparallel` can also compute approximations of the LR and Wald tests. The approximate LR test is computed by comparing the log likelihood from `ologit` or `oprobit` with the likelihoods obtained by pooling $J - 1$ binary models fit with `logit` or `probit` and making an adjustment for the correlation between the binary outcomes defined by $y \leq m$ (Wolfe and Gould 1998). The approximate Wald test, known as the Brant test, compares the estimates from binary logit models. Details on how this test is computed are found in Brant (1990) and Long (1997, 143-144). The `oparallel` command does not test for violations of parallel regressions for individual variables, so below we discuss the `brant` command, which does.

While the null hypothesis might be rejected because the β_m 's differ by m , Brant (1990) notes that the null hypothesis could be rejected because of other departures from the specified model. Reiterating this point, Greene and Hensher (2010) suggest that tests of the parallel assumption are only useful for "supporting or casting doubt on the basic model", but the tests do not indicate what the appropriate model might be. This issue is considered further in chapter 8.

3. `brant` is installed as part of the `spost13.ado` package.

7.6.1 Testing the parallel regression assumption using `oparallel`

`oparallel` can be used after `ologit` to compute the omnibus tests described above. The `ic` option provides the statistics AIC and BIC comparing the generalized ordered logit model with the ordered logit model.

```
. use gssclass4, clear
(gssclass4.dta | GSS Subjective Class Identification | 2013-11-20)
. ologit class i.female i.white i.year i.educ c.age##c.age income, nolog
(output omitted)
. oparallel, ic
```

Tests of the parallel regression assumption

	Chi2	df	P>Chi2
Wolfe Gould	328	18	0.000
Brant	243.4	18	0.000
score	257.8	18	0.000
likelihood ratio	328.4	18	0.000
Wald	258.1	18	0.000

Information criteria

	ologit	gologit	difference
AIC	10056.42	9764.03	292.39
BIC	10136.03	9963.05	172.98

The results labeled `likelihood ratio` and `Wald` are the LR and Wald tests based on the generalized ordered logit model. The line `Wolfe Gould` contains the approximate LR test, `Brant` refers to the Brant test, and `score` is the score test. All tests reject the null hypothesis with $p < 0.001$. The score and Wald tests have similar values, while the two LR tests are larger. We find that these tests are often, perhaps usually, significant.

The AIC and BIC statistics can be used to evaluate the trade-off between the better fit of the generalized model and the loss of parsimony from having $J - 1$ coefficients for each independent variable instead of just one. In this example, the smaller values of both the AIC and BIC statistics for `gologit` compared with `ologit` provide evidence against the `ologit` model compared with the model in which the parallel regression assumption is relaxed. It is common for the BIC statistic to prefer the `ologit` model even when the significance tests reject the parallel regression assumption, and this sometimes happens with AIC as well.

Although `oparallel` can be used with `ologit` but not with `oprobit`, the approximate LR test presented as `Wolfe Gould` can be performed with `oprobit` by using the user-written command `omodel`. `omodel`, however, does not support factor-variable notation.

7.6.2 Testing the parallel regression assumption using brant

The `SPost` command `brant` also computes the Brant test for the ORM. The advantage of our command, which is used by `oparallel` to make its computations, is that it provides separate tests for each of the independent variables in the model. After running `ologit`, you run `brant`, which has the following syntax:

```
brant [, detail]
```

The `detail` option provides a table of coefficients from each of the binary models. For example,

```
. brant, detail
```

Estimated coefficients from binary logits

Variable	y_gt_1	y_gt_2	y_gt_3
female			
female	-0.103	0.075	0.036
	-0.88	1.24	0.23
white			
white	-0.025	0.245	-0.155
	-0.19	3.04	-0.71
year			
1996	-0.170	-0.090	0.002
	-1.05	-1.20	0.01
2012	-0.749	-0.356	-0.686
	-4.64	-4.21	-2.97
educ			
hs only	0.259	0.277	-0.409
	2.00	3.22	-1.55
college	1.259	1.542	0.620
	4.67	14.50	2.33
age			
	-0.070	-0.050	-0.011
	-3.84	-4.86	-0.40
c.age#c.age			
	0.001	0.001	0.000
	4.01	7.24	1.15
income			
	0.050	0.011	0.011
	14.44	16.93	12.10
_cons			
	2.407	-0.932	-4.413
	5.34	-3.76	-6.19

legend: b/t

Brant test of parallel regression assumption

	chi2	p>chi2	df
All	243.39	0.000	18
1.female	2.16	0.339	2
1.white	6.52	0.038	2
2.year	0.47	0.789	2
3.year	7.03	0.030	2
2.educ	6.80	0.033	2
3.educ	12.42	0.002	2
age	3.28	0.194	2
c.age#c.age	2.69	0.261	2
income	125.59	0.000	2

A significant test statistic provides evidence that the parallel regression assumption has been violated.

The largest violation is for income, which may indicate particular problems with the parallel regression assumption for this variable. Looking at the coefficients from the binary logits, we see that for income the estimates from the binary logit of lower class versus working/middle/upper class differ from the other two binary logits. This suggests that income differences matter more for whether people report themselves as lower class than it does for either of the other thresholds. If the focus of our project was the relationship between income and subjective class identification, this would serve as a substantively interesting finding that we would have missed had we not used `brant`.

7.6.3 Caveat regarding the parallel regression assumption

In the majority of the real-world applications of the ORM that we have seen, the hypothesis of parallel regressions is rejected. Keep in mind, however, the tests of the parallel regression assumption are sensitive to other types of misspecification. Further, we have seen examples where the parallel regression assumption is violated but the predictions from `ologit` are very similar to those from the generalized ordered logit model or the multinomial logit model. When the hypothesis is rejected, consider alternative models that do not impose the constraint of parallel regressions. As illustrated in chapter 8, fitting the multinomial logit model on a seemingly ordinal outcome can lead to quite different conclusions. The generalized ordered logit model is another alternative to consider. Violation of the parallel regression assumption is not, however, a rationale for using the LRM. The assumptions implied by the application of the LRM to ordinal data are even stronger.

7.7 Overview of interpretation

Most of the rest of the chapter focuses on interpreting results from the ORM. First, we consider methods of interpretation that are based on transforming the coefficients. If the idea of a latent variable makes substantive sense, or if you are tempted to run a lin-

ear regression on an ordinal outcome, interpretations based on rescaling y^* to compute standardized coefficients can be used just like coefficients for the LRM. Coefficients can also be exponentiated and interpreted as odds ratios in the ordered logit model. We examine these strategies of interpretation first because they follow most straightforwardly from the methods of interpretation many readers are already familiar with from linear regression, but we regard them also as having important limitations that we discuss.

We then consider approaches to interpretation that use predicted probabilities, extending each of the methods for the BRM to multiple outcomes. We typically find these approaches far more informative. Because the ORM is nonlinear in the outcome probabilities, no approach can fully describe the relationship between a variable and the outcome probabilities. Consequently, you should consider each of these methods before deciding which approach is most effective in your application. As with models for binary outcomes, the basic command for interpretations based on predictions is `margins`, although our `mtable`, `mchange`, and `mgen` commands make things much simpler. Not only do these commands have the advantages illustrated for binary models in chapter 6, but when there are multiple outcome categories, `margins` can only compute predictions for one outcome at a time. Our commands will compute predictions for all categories and combine the results.

7.8 Interpreting transformed coefficients

As with the BRM, coefficients for the ordered logit model are about 1.7 times larger than those for the ordered probit model because of the arbitrary assumption about the variance of the error term. For this reason, neither ordered logit nor ordered probit coefficients offer a direct interpretation that is readily meaningful. There are two ways we can transform the coefficients into more meaningful quantities: standardization and odds ratios. In both cases, these interpretations are permissible only when the independent variable is not specified using polynomial or interaction terms.

7.8.1 Marginal change in y^*

In the ORM, $y^* = \mathbf{x}\beta + \varepsilon$, and the marginal change in y^* with respect to x_k is

$$\frac{\partial y^*}{\partial x_k} = \beta_k$$

Because y^* is latent, its true metric is unknown. The value of y^* depends on the identification assumption we make about the variance of the errors. As a result, the marginal change in y^* cannot be interpreted without standardizing by the estimated standard deviation of y^* , which is computed as

$$\hat{\sigma}_{y^*}^2 = \hat{\beta}' \widehat{\text{Var}}(\mathbf{x}) \hat{\beta} + \text{Var}(\varepsilon)$$

where $\widehat{\text{Var}}(\mathbf{x})$ is the covariance matrix for the observed x 's, $\widehat{\beta}$ contains maximum likelihood estimates, and $\text{Var}(\varepsilon) = 1$ for ordered probit and $\pi^2/3$ for ordered logit. Then the y^* -standardized coefficient for x_k is

$$\beta_k^{Sy^*} = \frac{\beta_k}{\sigma_{y^*}}$$

which can be interpreted as follows:

For a unit increase in x_k , y^* is expected to increase by $\beta_k^{Sy^*}$ standard deviations, holding all other variables constant.

The fully standardized coefficient is

$$\beta_k^S = \frac{\sigma_k \beta_k}{\sigma_{y^*}}$$

which can be interpreted as follows:

For a standard deviation increase in x_k , y^* is expected to increase by β_k^S standard deviations, holding all other variables constant.

These coefficients are computed using `listcoef` with the `std` option. For example, after fitting the ordered logit model,

```
. ologit class i.female i.white i.year i.educ c.age#c.age income, nolog
(output omitted)
. listcoef, std help
ologit (N=5620): Unstandardized and standardized estimates
Observed SD: 0.6749
Latent SD: 2.1437
```

	b	z	P> z	bStdX	bStdY	bStdXY	SDofX
female							
female	0.0162	0.298	0.765	0.008	0.008	0.004	0.498
white							
white	0.2363	3.277	0.001	0.092	0.110	0.043	0.389
year							
1996	-0.0799	-1.158	0.247	-0.040	-0.037	-0.019	0.498
2012	-0.5039	-6.594	0.000	-0.233	-0.235	-0.109	0.463
educ							
hs only	0.3705	4.730	0.000	0.183	0.173	0.085	0.493
college	1.5656	15.994	0.000	0.670	0.730	0.313	0.428
age							
age	-0.0488	-5.308	0.000	-0.825	-0.023	-0.385	16.897
c.age#c.age							
c.age#c.age	0.0007	7.646	0.000	1.202	0.000	0.561	1695.148
income							
income	0.0116	22.203	0.000	0.770	0.005	0.359	66.258

```
b = raw coefficient
z = z-score for test of b=0
P>|z| = p-value for z-test
bStdX = x-standardized coefficient
bStdY = y-standardized coefficient
bStdXY = fully standardized coefficient
SDofX = standard deviation of X
```

In our example, we can think of the dependent variable as measuring subjective social standing. Consequently, examples of interpretation are as follows:

The subjective social standing of those with a high school degree as their highest degree is 0.17 standard deviations higher than that of those who do not have a high school diploma, holding all other variables constant.

Each standard deviation increase in household income increases support by 0.36 standard deviations, holding all other variables constant.

Because the coefficients in the columns **b** and **bStdX** are not based on standardizing y^* , they should not be interpreted. And while **listcoef** presents coefficients for **age**, these should not be interpreted because you cannot change age while holding age-squared constant. An advantage of the methods of interpretation using probabilities that we discuss later in the chapter is that they can be used more simply when polynomials or interactions are in the model.

Although we do not often use coefficients for the marginal change in y^* to interpret the ORM, we believe that this is a much better approach than fitting the LRM with an ordinal dependent variable and interpreting the LRM coefficients.

7.8.2 Odds ratios

The ordinal logit model (but not the ordinal probit model) can also be interpreted using odds ratios. Equation (7.2) defined the ordered logit model as

$$\Omega_{\leq m | > m}(\mathbf{x}) = \exp(\tau_m - \mathbf{x}\beta)$$

For example, with four outcomes we would simultaneously estimate the three equations

$$\Omega_{\leq 1 | > 1}(\mathbf{x}) = \exp(\tau_1 - \mathbf{x}\beta)$$

$$\Omega_{\leq 2 | > 2}(\mathbf{x}) = \exp(\tau_2 - \mathbf{x}\beta)$$

$$\Omega_{\leq 3 | > 3}(\mathbf{x}) = \exp(\tau_3 - \mathbf{x}\beta)$$

Using the same approach as shown for binary logit, the effect of a unit change in x_k equals

$$\frac{\Omega_{\leq m | > m}(\mathbf{x}, x_k + 1)}{\Omega_{\leq m | > m}(\mathbf{x}, x_k)} = e^{-\beta_k} = \frac{1}{e^{\beta_k}}$$

The value of the odds ratio does not depend on the value of m , which is why the parallel regression assumption is also known as the proportional-odds assumption. We could interpret the odds ratio as follows:

For a unit increase in x_k , the odds of a lower outcome compared with a higher outcome are changed by the factor $\exp(-\beta_k)$, holding all other variables constant.

For a change in x_k of δ ,

$$\frac{\Omega_{\leq m | > m}(\mathbf{x}, x_k + \delta)}{\Omega_{\leq m | > m}(\mathbf{x}, x_k)} = \exp(-\delta \times \beta_k) = \frac{1}{\exp(\delta \times \beta_k)}$$

which we interpret as follows:

For an increase of δ in x_k , the odds of a lower outcome compared with a higher outcome change by a factor of $\exp(-\delta \times \beta_k)$, holding all other variables constant.

Notice that the odds ratio is derived by changing one variable, x_k , while holding all other variables constant. Accordingly, you do not want to compute the odds ratio for a variable that is included as a polynomial (for example, age and age-squared) or is included in an interaction term.

The odds ratios for a unit and a standard deviation change of the independent variables can be computed with `listcoef`, which lists the factor changes in the odds of higher versus lower outcomes. You could also obtain odds ratios by using the `or` option with the `ologit` command. Here we request odds ratios for only `white`, `year`, `income`, and `age`:

```
. listcoef white year income age, help
ologit (N=5620): Factor change in odds
Odds of: >m vs <=m
```

	b	z	P> z	e ^b	e ^b StdX	SDofX
white						
white	0.2363	3.277	0.001	1.267	1.096	0.389
year						
1996	-0.0799	-1.158	0.247	0.923	0.961	0.498
2012	-0.5039	-6.594	0.000	0.604	0.792	0.463
age	-0.0488	-5.308	0.000	0.952	0.438	16.897
c.age#c.age	0.0007	7.646	0.000	1.001	3.328	1695.148
income	0.0116	22.203	0.000	1.012	2.160	66.258

```
b = raw coefficient
z = z-score for test of b=0
P>|z| = p-value for z-test
eb = exp(b) = factor change in odds for unit increase in X
ebStdX = exp(b*SD of X) = change in odds for SD increase in X
SDofX = standard deviation of X
```

Here are some interpretations:

The odds of reporting higher subjective class standing are 0.60 times smaller in 2012 than they were in 1980, holding all other variables constant.

For a standard deviation increase in income, the odds of indicating higher social standing increase by a factor of 2.16, holding all other variables constant.

Although odds ratios for `age` are shown, these should not be interpreted because you cannot change age while holding age-squared constant. If you prefer, you can compute coefficients for the percentage change in the odds by adding the `percent` option:


```
. listcoef white year income, percent
ologit (N=5620): Percentage change in odds
Odds of: >m vs <=m
```

	b	z	P> z	%	%StdX	SDofX
white						
white	0.2363	3.277	0.001	26.7	9.6	0.389
year						
1996	-0.0799	-1.158	0.247	-7.7	-3.9	0.498
2012	-0.5039	-6.594	0.000	-39.6	-20.8	0.463
income	0.0116	22.203	0.000	1.2	116.0	66.258

These results can be interpreted as follows:

The odds of reporting higher subjective class standing are 40% smaller in 2012 than they were in 1980, holding all other variables constant.

For a standard deviation increase in income, the odds of indicating higher social standing increase by 116%, holding all other variables constant.

So far, we interpreted the factor changes in the odds of lower outcomes compared with higher outcomes. This is done because the model is traditionally written in terms of the odds of lower versus higher outcomes, $\Omega_{\leq m | > m}(\mathbf{x})$, leading to the factor change coefficient of $\exp(-\beta_k)$. We could just as well consider the factor change in the odds of higher versus lower values; that is, changes in the odds $\Omega_{> m | \leq m}(\mathbf{x})$, which equals $\exp(\beta_k)$. These odds ratios can be obtained by adding the option `reverse`:

```
. listcoef year, reverse
ologit (N=5620): Factor change in odds
Odds of: <=m vs >m
```

	b	z	P> z	e ^b	e ^b StdX	SDofX
year						
1996	-0.0799	-1.158	0.247	1.083	1.041	0.498
2012	-0.5039	-6.594	0.000	1.655	1.262	0.463

Notice that the output now says Odds of: $\leq m$ vs $> m$ instead of Odds of: $> m$ vs $\leq m$, as it did earlier. This factor change of 1.66 for 2012 is the inverse of the earlier value 0.60. Our interpretation is the following:

The odds of reporting lower social standing are about 1.66 times larger in 2012 than they were in 1980, holding all other variables constant.

When presenting odds ratios, some people find it easier to understand the results if you talk about increases in the odds rather than decreases. That is, it is clearer to say,

"The odds increased by a factor of 2" than to say, "The odds decreased by a factor of 0.5". If you agree, then you can reverse the order when presenting odds.

When interpreting odds ratios, remember three points that were discussed in detail in chapter 6. First, because odds ratios are multiplicative coefficients, positive and negative effects should be compared by taking the inverse of the negative effect (or vice versa). For example, a negative factor change of 0.5 has the same magnitude as a positive factor change of $2 = 1/0.5$. Second, interpretation assumes only that the other variables have been held constant, not held at specific values. Third, a constant factor change in the odds does not correspond to a constant change or constant factor change in the probability.

As with binary outcomes, we discuss odds ratios because they are commonly used with these models and provide a compact means of interpretation. Yet we think they are overused, especially in data based on population samples instead of case-control studies. The meaning of the magnitude of multiplicative changes in odds is often unclear to audiences, perhaps even more so when thinking about transitions across thresholds that divide sets of categories. We are perhaps inclined to favor y -standardized coefficients over odds ratios for ordinal outcomes; the idea of a standard deviation change in the latent variable allows results to be understood more clearly because of the analogue to linear regression. Better still, however, are methods of interpretation that are based on predicted probabilities, which we discuss next.

7.9 Interpretations based on predicted probabilities

As noted, we usually prefer interpretations based on predicted probabilities. We find these interpretations to be both clearer for our own thinking and more effective with audiences. Probabilities can be estimated with the formula

$$\widehat{\Pr}(y = m | \mathbf{x}) = F(\hat{\tau}_m - \mathbf{x}\hat{\beta}) - F(\hat{\tau}_{m-1} - \mathbf{x}\hat{\beta})$$

Cumulative probabilities are computed as

$$\widehat{\Pr}(y \leq m | \mathbf{x}) = \sum_{k \leq m} \widehat{\Pr}(y = k | \mathbf{x}) = F(\hat{\tau}_m - \mathbf{x}\hat{\beta})$$

The values of \mathbf{x} can be based on observations in the sample or can be hypothetical values of interest.

The following sections use predicted probabilities in a variety of ways. We begin by examining the distribution of predictions for each observation in the estimation sample as a first step in evaluating your model. Next, we show how marginal effects provide an overall assessment of the impact of each variable. To focus on particular types of respondents, we compute predictions for ideal types defined by substantively motivated characteristics for all independent variables. Extending methods from chapter 6, we show how to statistically test differences in the predictions between ideal types. For categorical predictors, tables of predictions computed as these variables change is an

effective way to demonstrate the effects of these variables. An important challenge is to decide where to hold the values of other variables when making predictions. Finally, we plot predictions as a continuous independent variable changes.

7.10 Predicted probabilities with predict

After fitting a model with `ologit` or `oprobit`, a useful first step for assessing your model is to compute the in-sample predictions with the command

```
predict newvar1[newvar2[newvar3...]] [if] [in]
```

where you specify one new variable name for each category of the dependent variable. For instance, in the following example, `predict` specifies that the variables `prlower`, `prworking`, `prmiddle`, and `prupper` be created with predicted values for the four outcome categories:

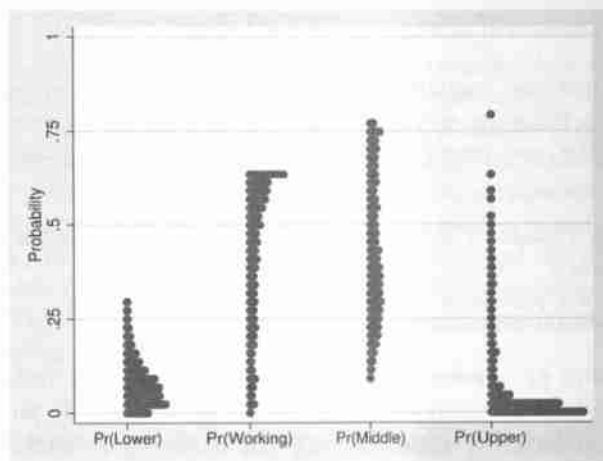
```
. ologit class i.female i.white i.year i.educ c.age#c.age income, nolog
      (output omitted)
. predict prlower prworking prmiddle prupper
      (option pr assumed; predicted probabilities)
```

The message (option `pr` assumed; predicted probabilities) reflects that `predict` can compute many different quantities. Because we did not specify an option indicating which quantity to predict, the default option `pr` for predicted probabilities was assumed.

Predictions in the sample are useful for getting a general sense of what is going on in your model and can be useful for uncovering problems in your data. For example, if there are observations where the predicted probability of being in `prlower` (or any other outcome) are noticeably larger or smaller than the other predictions, you might check whether there are data problems for those observations. The range of predictions can also give you a rough idea of how large marginal effects can be for a given outcome. If the range of probabilities is small within the estimation sample, the effects of the independent variables will also be small. If the distribution of predictions has multiple modes—let's say, two—it suggests there could be a binary predictor that is important. Although sometimes the distribution of predictions leads to additional data checking or model revision, often it only assures you that predictions are reasonable and that you are ready for the methods of interpretation that we consider in the remainder of this chapter.

An easy way to see the distribution of the predictions is with `dotplot`, one of our favorite commands for quickly checking data:

```
. label var prlower "Pr(Lower)"
. label var prworking "Pr(Working)"
. label var prmiddle "Pr(Middle)"
. label var prupper "Pr(Upper)"
. dotplot prlower prworking prmiddle prupper,
> ylabel(0(.25)1, grid gmin gmax) ytitle("Probability")
```



The predicted probabilities for the extreme categories of lower and upper tend to be less than 0.20, with most predictions for the middle categories falling between 0.25 and 0.75. The probabilities for the middle two categories are generally larger than the probabilities of the extreme categories, reflecting the higher observed proportions of observations for these categories. The long tail for the probabilities for “Upper” lead us to examine the data further, but no problems were found. If you look at the probabilities of identifying as working class, you will notice a spike in the number of cases near its highest predicted probability, around 0.65. This is common for middle categories when plotting predicted probabilities for the ORM and should not be cause for concern. For extreme categories, the predicted probabilities of individual observations in the ORM are bound only by 0 and 1. For middle categories, however, the distance between estimated cutpoints implies a maximum predicted probability, where a greater distance between cutpoints implies a higher maximum (see section 7.15 for more about why this is so).

In this example, with `predict` we specified separate variables for each outcome category. Because of this, `predict` understood that we wanted predicted probabilities for each category. Had we specified only one variable, `predict` would generate predicted values of y^* , not probabilities. To compute the predicted probability for a single outcome category, you need the `outcome(#)` option, such as `predict prmiddle, outcome(3)`. The `#` specified with `outcome(#)` is the rank position of the category from lowest to

highest. If the outcome variable is numbered with consecutive integers starting with 1, as in our example, then `#` corresponds to the outcome value. However, if our outcome values were numbered 0, 1, 2, and 3, then `outcome(1)` would provide the predicted probability that $y = 0$, not that $y = 1$. We find this extremely confusing in practice. To avoid it, we strongly recommend numbering outcome values with consecutive integers starting with 1 whenever working with ordinal or nominal outcomes.

Examining predicted probabilities within the sample provides a first, quick check of the model. To understand and present the substantive findings, however, you will usually want to compute predictions at specific, substantively informative values.

7.11 Marginal effects

The marginal change in the probability of outcome m is computed as

$$\frac{\partial \Pr(y = m | \mathbf{x})}{\partial x_k} = \frac{\partial F(\tau_m - \mathbf{x}\beta)}{\partial x_k} - \frac{\partial F(\tau_{m-1} - \mathbf{x}\beta)}{\partial x_k}$$

which is the slope of the curve relating x_k to $\Pr(y=m|\mathbf{x})$, holding all other variables constant. The value of the marginal change depends on the value of x_k where the change is evaluated, as well as the values of all other x 's. Because the marginal change can be misleading when the probability curve is changing rapidly, we usually prefer using discrete change. The discrete change is the change in the probability of m for a change in x_k from the start value x_k^{start} to the end value x_k^{end} (for example, a change from $x_k = 0$ to $x_k = 1$), holding all other x 's constant. Formally,

$$\frac{\Delta \Pr(y = m | \mathbf{x})}{\Delta x_k (x_k^{\text{start}} \rightarrow x_k^{\text{end}})} = \Pr(y = m | \mathbf{x}, x_k = x_k^{\text{end}}) - \Pr(y = m | \mathbf{x}, x_k = x_k^{\text{start}})$$

where $\Pr(y = m | \mathbf{x}, x_k)$ is the probability that $y = m$ given \mathbf{x} , noting a specific value for x_k . The change indicates that when x_k changes from x_k^{start} to x_k^{end} , the probability of outcome m changes by $\Delta \Pr(y = m | \mathbf{x}) / \Delta x_k$, holding all other variables at the specific values in \mathbf{x} . The magnitude of the discrete change depends on the value at which x_k starts, the amount of change in x_k , and the values of all other variables.

For both marginal and discrete change, we can compute average marginal effects (AMEs), marginal effects at the mean (MEMs), or marginal effects at representative values other than the means. As with the BRM, we find AMEs to be the most useful summary of the effects, thus we consider AMEs for the ORM in this section. MEMs are considered briefly in section 7.15. Examining the distribution of effects over observations is also valuable. To save space, we do not illustrate this in the current chapter, but this can be done using the commands presented in section 8.8.1.

To illustrate the use of marginal effects in ordinal models, we begin by examining the average marginal change for income. The marginal change can be computed with `mchange`, where we select the variable `income` and use `amount(marginal)` to request only marginal changes without any discrete changes. Because we have not included the `atmeans` option, `mchange` computes the AME over all observations:


```
. mchange income, amount(marginal)
```

ologit: Changes in Pr(y) | Number of obs = 5620

Expression: Pr(class), predict(outcome())

	lower	working	middle	upper
income				
Marginal	-0.001	-0.002	0.002	0.000
p-value	0.000	0.000	0.000	0.000
Average predictions				
Pr(y base)	0.071	0.460	0.434	0.034

The marginal changes are in the row labeled **Marginal**, with the significance level for the test of the hypothesis that the change is 0 listed in the row **p-value**. In this example, the marginal changes are all less than 0.003 in magnitude, but the *p*-values are nevertheless significant.⁴ We see that on average higher income decreases identification with lower and working class, while increasing identification with middle and upper class. Across all categories, the AMEs must sum to 0, because any increase in the probability of one category must be offset by a decrease in another category. The results in the output might not sum exactly to 0, however, because of rounding. The average predicted probabilities of each outcome are listed below the table of marginal changes. The average predicted probability of someone identifying as lower class is 0.07, as working class is 0.46, and so on. These probabilities must, of course, sum to 1.

In this example, the marginal changes with respect to income are small, and it is difficult to grasp how large the effects of income are in terms of changes in the probabilities of class identification. A marginal change is the instantaneous rate of change that does not correspond exactly to the amount of change in the probability for a change of one unit in the independent variable. If the probability curve is approximately linear where the change is evaluated, the marginal change will approximate the effect of a unit change in the variable on the probability of an outcome. The best way to determine how well the marginal change approximates the discrete change is to compute the discrete change, which we do next.

The variable **income** is measured in thousands of dollars. Looking at the descriptive statistics for this variable,

```
. sum income
```

Variable	Obs	Mean	Std. Dev.	Min	Max
income	5620	68.07737	66.25833	.51205	324.2425

we see that the range is over \$300,000, so a unit change is too small for describing an effect of income on class identification. Another way of thinking about this is that a \$1,000 difference in income is substantively small compared with what we might

4. If you wanted more decimal places shown for the changes, you could add the option `dec(6)`, for example.

anticipate would have an appreciable effect on whether people view themselves as, for example, working class rather than middle class. By default, `mchange` computes discrete changes for both a 1-unit change and a standard deviation change, where we use `brief` to suppress showing the average probabilities:

```
. mchange income, brief
ologit: Changes in Pr(y) | Number of obs = 5620
Expression: Pr(class), predict(outcome())
```

		lower	working	middle	upper
income					
	+1	-0.001	-0.002	0.002	0.000
	p-value	0.000	0.000	0.000	0.000
	+SD	-0.036	-0.119	0.126	0.030
	p-value	0.000	0.000	0.000	0.000
	Marginal	-0.001	-0.002	0.002	0.000
	p-value	0.000	0.000	0.000	0.000

We can interpret the results for a change of a standard deviation in income as follows:

On average, a standard deviation increase in income (about \$66,000) is associated with a 0.036 decrease in the probability of identifying as lower class and a 0.119 decrease in identifying as working class. This is offset by an increase of 0.126 in the probability of identifying as middle class and a 0.030 increase in upper-class identification. All effects are significant at the 0.001 level.

Instead of a standard deviation change, we might be interested in a change of a specific amount, like a \$25,000 increase. It might be tempting to (incorrectly) compute the discrete change for a 25-unit change in income by simply multiplying the 1-unit discrete change by 25. Although this will give you approximately the right answer if the probability curve is nearly linear over the range of the change, in some cases it can give misleading results and even the wrong sign. To be safe, do not do it! Instead, the `delta(#)` option for `mchange` computes the discrete change as an independent value changes from the base value to # units above the base value. Here we use `delta(25)`:

```
. mchange income, delta(25) amount(delta) brief
ologit: Changes in Pr(y) | Number of obs = 5620
Expression: Pr(class), predict(outcome())
```

		lower	working	middle	upper
income					
	+delta	-0.016	-0.042	0.049	0.009
	p-value	0.000	0.000	0.000	0.000

We can interpret the results as follows:

On average, a \$25,000 change in income is associated with a 0.049 increase in the probability of identifying as middle class and a 0.042 decrease in the probability of identifying as working class.

We can also compute changes in the predicted probability as a continuous variable changes from its minimum to the maximum by specifying the option `amount(range)`:

```
. mchange income, amount(range) brief
ologit: Changes in Pr(y) | Number of obs = 5620
Expression: Pr(class), predict(outcome())
```

		lower	working	middle	upper
income					
	Range	-0.112	-0.514	0.371	0.255
	p-value	0.000	0.000	0.000	0.000

With a variable like `income` where there might be a few respondents with very high incomes, a trimmed range might be more informative. Here, by specifying `trim(5)`, we examine the effect of a change from the 5th percentile of income to the 95th percentile:⁵

```
. mchange income, amount(range) trim(5) brief
ologit: Changes in Pr(y) | Number of obs = 5620
Expression: Pr(class), predict(outcome())
```

		lower	working	middle	upper
income					
	5% to 95%	-0.094	-0.365	0.385	0.074
	p-value	0.000	0.000	0.000	0.000

The effects are substantially smaller, most noticeably for those identifying as upper class. If you look carefully, you will notice that as a result of decreasing the amount of change in income, the change in the probability of middle-class affiliation is increasing. The reason for this apparent anomaly is explained in section 7.15.

The `mchange` command uses `margins` to compute the changes. If you want the convenience of `mchange` but also want to see how `margins` is being used, you can specify the option `commands` to see the `margins` commands used by `mchange` or specify the `detail` option to obtain the full `margins` output.

7.11.1 Plotting marginal effects

As we suggested for the BRM, the AME is a valuable tool for examining the effects of your variables, and we often compute these effects for an initial review of the results of a model. Without doubt, AMEs are far more informative than the parameter estimates or

5. `mchange` does not present the values of income at the 5th and 95th percentiles, but these can be easily computed using the `summarize` command with the `detail` option.

odds ratios. There is, however, a lot of information to be absorbed. By default, for each continuous variable, `mchange` computes the marginal change and discrete changes for a 1-unit and a standard deviation change in continuous variables; for factor variables, `mchange` computes a discrete change from 0 to 1. One way to limit the amount of information is to only look at discrete changes of a standard deviation for continuous variables. For example,

```
. ologit class i.female i.white i.year i.educ c.age##c.age income, nolog
(output omitted)
. mchange, amount(sd) brief
ologit: Changes in Pr(y) | Number of obs = 5620
Expression: Pr(class), predict(outcome())
```

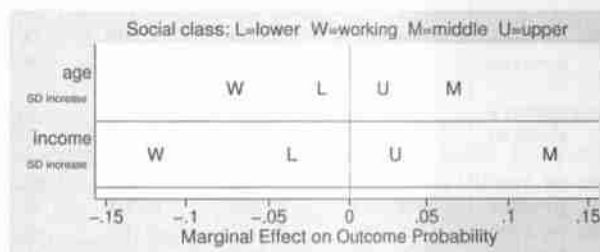
		lower	working	middle	upper
female					
	female vs male	-0.001	-0.002	0.003	0.000
	p-value	0.766	0.765	0.765	0.765
white					
	white vs nonwhite	-0.016	-0.031	0.041	0.006
	p-value	0.002	0.001	0.001	0.001
year					
	1996 vs 1980	0.004	0.012	-0.014	-0.003
	p-value	0.243	0.249	0.246	0.253
	2012 vs 1980	0.033	0.067	-0.086	-0.014
	p-value	0.000	0.000	0.000	0.000
	2012 vs 1996	0.029	0.055	-0.073	-0.011
	p-value	0.000	0.000	0.000	0.000
educ					
	hs only vs not hs grad	-0.029	-0.047	0.070	0.006
	p-value	0.000	0.000	0.000	0.000
	college vs not hs grad	-0.079	-0.252	0.286	0.046
	p-value	0.000	0.000	0.000	0.000
	college vs hs only	-0.050	-0.205	0.216	0.039
	p-value	0.000	0.000	0.000	0.000
age					
	+SD	-0.018	-0.071	0.067	0.022
	p-value	0.000	0.000	0.000	0.000
income					
	+SD	-0.036	-0.119	0.126	0.030
	p-value	0.000	0.000	0.000	0.000

Even so, there are a lot of coefficients. Fortunately, they can be quickly understood by plotting them. To explain how to do this, we begin by examining the AMEs for a standard deviation change in income and age. Because the model includes age and age-squared, when age is increased by a standard deviation, we need to increase age-squared by the appropriate amount. This is done automatically by Stata because we entered age into the model with the factor-variable notation `c.age##c.age`. To compute the average discrete changes for a standard deviation increase, type


```
. mchange age income, amount(sd) brief
ologit: Changes in Pr(y) | Number of obs = 5620
Expression: Pr(class), predict(outcome())
```

		lower	working	middle	upper
age	+SD	-0.018	-0.071	0.067	0.022
	p-value	0.000	0.000	0.000	0.000
income	+SD	-0.036	-0.119	0.126	0.030
	p-value	0.000	0.000	0.000	0.000

`mchange` leaves these results in memory, and they are used by our `mchangeplot` command to create the plot.



The horizontal axis indicates the magnitude of the effect, with the letters within the graph marking the discrete change for each outcome. For example, the M in the row for income shows that, on average for a standard deviation change in income, the probability of identifying with the middle class increases by 0.126. Overall, it is apparent that the effects of income are larger than those for a standard deviation change in age. For both variables, the effects are in the same directions with the same relative magnitudes. (Before proceeding, you should make sure you see how the graph corresponds to the output from `mchange` above.)

The plot was produced with the following command:

```
mchangeplot age income, symbols(L W M U) min(-.15) max(.15) gap(.05) ///
    title("Social class: L=lower W=working M=middle U=upper", ///
    size(medsmall)) ysize(1.3) scale(2.1)
```

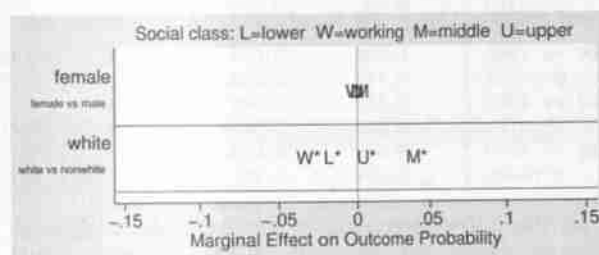
After the variables are selected, `symbols()` specifies the letters to use for each outcome. By default, the first letters in the value labels are used, but here we chose to use capital letters instead of the lowercase letters used by `class`'s value labels. The options `min()`, `max()`, and `gap()` define the tick marks and labels on the x axis. The `ysize()` and `scale()` options affect the size of the graph and the scaled font size. Details on all options for `mchangeplot` can be found by typing `help mchangeplot`.

Next, we consider the discrete change for a change from 0 to 1 for the binary variables `female` and `white`:


```
. mchange female white, brief
ologit: Changes in Pr(y) | Number of obs = 5620
Expression: Pr(class), predict(outcome())
```

	lower	working	middle	upper
female				
female vs male	-0.001	-0.002	0.003	0.000
p-value	0.766	0.765	0.765	0.765
white				
white vs nonwhite	-0.016	-0.031	0.041	0.006
p-value	0.002	0.001	0.001	0.001

When producing the graph, we use the option `sig(.05)` to add an asterisk (*) to effects that are significant at the 0.05 level:



The effects of being female are small and nonsignificant, which is expected given that the coefficient for `female` is not significant. For the contrast between whites and nonwhites, on the other hand, we see significant differences, which we interpret as follows:

The predicted probability of identifying as middle class is on average 0.04 higher for a white person than for an otherwise similar nonwhite person, while the predicted probability of identifying as working class is 0.03 lower.

For factor variables that have more than two categories, we want to examine the contrasts between all categories. Consider variables `year` and `educ`, each of which have three categories:

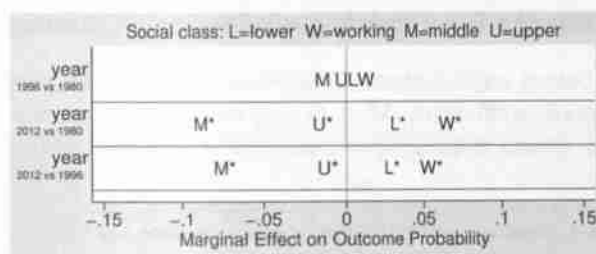
```
. mchange educ year, brief
ologit: Changes in Pr(y) | Number of obs = 5620
Expression: Pr(class), predict(outcome())
```

	lower	working	middle	upper
educ				
hs only vs not hs grad	-0.029	-0.047	0.070	0.006
p-value	0.000	0.000	0.000	0.000
college vs not hs grad	-0.079	-0.252	0.286	0.046
p-value	0.000	0.000	0.000	0.000
college vs hs only	-0.050	-0.205	0.216	0.039
p-value	0.000	0.000	0.000	0.000
year				
1996 vs 1980	0.004	0.012	-0.014	-0.003
p-value	0.243	0.249	0.246	0.253
2012 vs 1980	0.033	0.067	-0.086	-0.014
p-value	0.000	0.000	0.000	0.000
2012 vs 1996	0.029	0.055	-0.073	-0.011
p-value	0.000	0.000	0.000	0.000

`mchange` computes all the pairwise contrasts. For example, with `year` the output compares those answering the survey in 1996 with those in 1980, in 2012 with 1980, and in 2012 with 1996. One of the contrasts is redundant in the sense that it can be computed from the other two. For example, looking at lower-class identity, the change in probability of 0.004 from 1980 to 1996 plus the change of 0.029 from 1996 to 2012 equals the change of 0.033 from 1980 to 2012. Still, it is often useful to examine all contrasts to find patterns. For this, we find that plotting the effects works well:

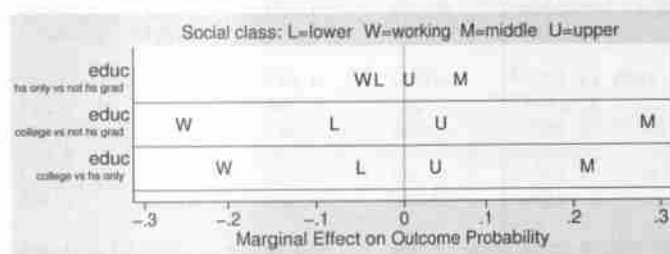
```
mchangeplot year, ///
  symbols(L W M U) min(-.15) max(.15) gap(.05) ///
  sig(.05) leftmargin(5) ///
  title("Social class: L=lower W=working M=middle U=upper", ///
  size(medsmall)) ysize(1.3) scale(2.1)
```

The `significance()` option specifies that *'s should be added to the plot symbols if the effect is significant at the given level—in this case, 0.05. The `leftmargin()` option increases the left margin of the graph to accommodate the value labels used with factor variables. The argument 5 is the percentage of the graph size to be added to the left. The resulting graph looks like this:



It is immediately apparent that there was little change from 1980 to 1996, while much larger and statistically significant changes occurred in 2012 compared with either of the earlier survey years.

Next, we plot the effects for `educ` without the `sig()` option because all the effects are significant:



These effects are larger than those for the year of the survey (notice that the x scale is not the same in the two graphs). Even though effects are statistically significant, the largest effects are comparing those who have a college degree with others, regardless of whether they have a high school diploma or did not graduate. With the overall pattern in mind, we can interpret multiple contrasts for a single category as follows:

On average, having graduated from college increases a person's probability of identifying as middle class by 0.22 compared with having graduated only from high school, and by 0.29 compared with not having graduated from high school.

Alternatively, we could interpret multiple categories for the same contrast:

Compared with those who have graduated only from high school, we find that graduating from college on average increases the probability of identifying as upper class by 0.04 and of identifying as middle class by 0.22, while the probability of identifying as working class decreases by 0.21 and of identifying as lower class by 0.05.

7.11.2 Marginal effects for a quick overview

AMES are a much better way to obtain a quick overview of the magnitudes of effects than are the estimated coefficients. After fitting your model, you can obtain a table of all effects by simply typing `mchange`, perhaps restricting effects to discrete changes of a standard deviation:

```
. ologit class i.female i.white i.year i.educ c.age##c.age income, nolog
(output omitted)
```

```
. mchange, amount(sd) brief
```

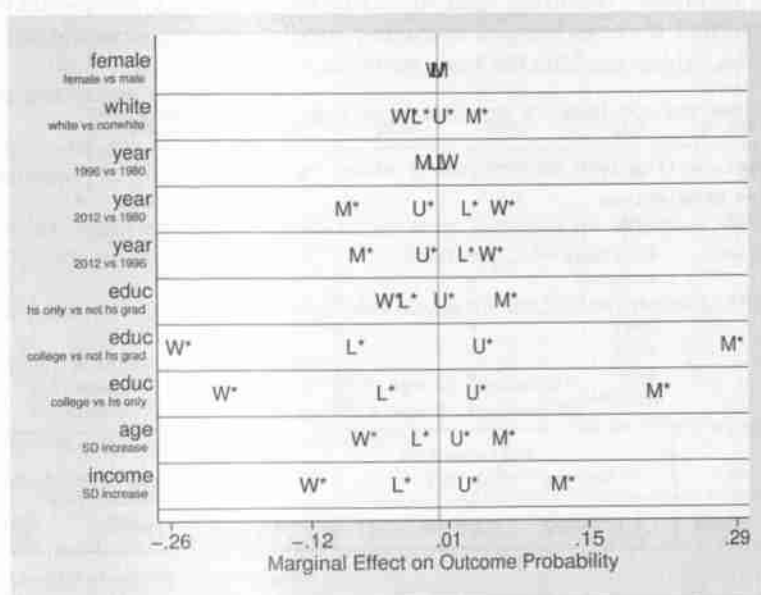
```
ologit: Changes in Pr(y) | Number of obs = 5620
```

```
Expression: Pr(class), predict(outcome())
```

		lower	working	middle	upper
female					
	female vs male	-0.001	-0.002	0.003	0.000
	p-value	0.766	0.765	0.765	0.765
white					
	white vs nonwhite	-0.016	-0.031	0.041	0.006
	p-value	0.002	0.001	0.001	0.001
year					
	1996 vs 1980	0.004	0.012	-0.014	-0.003
	p-value	0.243	0.249	0.246	0.253
	2012 vs 1980	0.033	0.067	-0.086	-0.014
	p-value	0.000	0.000	0.000	0.000
	2012 vs 1996	0.029	0.055	-0.073	-0.011
	p-value	0.000	0.000	0.000	0.000
educ					
	hs only vs not hs grad	-0.029	-0.047	0.070	0.006
	p-value	0.000	0.000	0.000	0.000
	college vs not hs grad	-0.079	-0.252	0.286	0.046
	p-value	0.000	0.000	0.000	0.000
	college vs hs only	-0.050	-0.205	0.216	0.039
	p-value	0.000	0.000	0.000	0.000
age					
	+SD	-0.018	-0.071	0.067	0.022
	p-value	0.000	0.000	0.000	0.000
income					
	+SD	-0.036	-0.119	0.126	0.030
	p-value	0.000	0.000	0.000	0.000

With a simple command, you can plot the effects:

```
. mchangeplot, symbols(L W M U) sig(.05) leftmargin(5)
```



A quick review highlights which variables we might want to examine more closely.

7.12 Predicted probabilities for ideal types

Ideal types define substantively interesting cases in the data by specifying values of the independent variables. Predicted probabilities for these types of individuals (or whatever the unit of analysis may be) can be computed with `mtable` or `margins`. Unlike marginal effects, by comparing two or more ideal types, you can compare probabilities as a whole set of independent variables vary, not just a change in a single variable.

In our example, ideal types can be used to examine what more and less advantaged individuals looks like and how they differ in their class identification. For instance, we might want to compare the following hypothetical individuals surveyed in 2012:

- A 25-year-old, nonwhite man without a high school diploma and with a household income of \$30,000 per year.
- A 60-year-old, white woman with a college degree and with a household income of \$150,000 per year.

To compute the predictions, we begin by using `margins` before showing how `mtable` can simplify the work. We use `at()` to specify values of the independent variables. If

there are variables whose values are not specified with `at()`, we can use the option `atmeans` to assign them to their means. Otherwise, by default, `margins` and `mtable` would compute the average predicted probability over the sample for the unspecified independent variables. We do not want to do this because ideal types should be thought of as hypothetical observations, so averaging predictions over observations for some independent variables muddles the interpretation.

Using values we specified for our first ideal type, we run `margins`:

```
. margins, at(female=0 white=0 year=3 educ=1 age=25 income=30)
Adjusted predictions      Number of obs   =      5620
Model VCE      : OIM
Expression      : Pr(class==1), predict()
at              : female          =          0
                  white           =          0
                  year            =          3
                  educ            =          1
                  age             =         25
                  income          =         30
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
._cons	.2300032	.0202718	11.35	0.000	.1902712	.2697352

`margins` can only compute a prediction for a single outcome. Because we did not specify which outcome, `margins` used the default prediction, which is described as `Pr(class==1), predict()`. This is the predicted probability for the first outcome. Hence, we find that the predicted probability of identifying as lower class for our first ideal type is 0.23.

To compute probabilities for other outcomes, we use the `predict(outcome(#))` option, where `#` is the value of the outcome for which we want a prediction. To compute the probabilities for all values of `class`, we must run four `margins` commands that vary the outcome value:

```
margins, at(female=0 white=0 year=3 educ=1 age=25 income=30) ///
  predict(outcome(1))
margins, at(female=0 white=0 year=3 educ=1 age=25 income=30) ///
  predict(outcome(2))
margins, at(female=0 white=0 year=3 educ=1 age=25 income=30) ///
  predict(outcome(3))
margins, at(female=0 white=0 year=3 educ=1 age=25 income=30) ///
  predict(outcome(4))
```

It is easier, however, to use `mtable`, which computes predictions for all outcome categories and combines them into a single table. The option `ci` indicates that we want the output to show the confidence interval.


```
. mtable, at(female=0 white=0 year=3 educ=1 age=25 income=30) ci
```

```
Expression: Pr(class), predict(outcome())
```

	lower	working	middle	upper
Pr(y)	0.230	0.634	0.133	0.003
ll	0.190	0.613	0.108	0.002
ul	0.270	0.655	0.159	0.004

```
Specified values of covariates
```

	female	white	year	educ	age	income
Current	0	0	3	1	25	30

The results for category *lower* match those from *margins* above, plus we have predictions for the other outcomes.

We could also compute predicted probabilities for both ideal types at the same time:

```
. mtable, atright norownum width(7)
```

```
> at(female=0 white=0 year=3 ed=1 age=25 income=30)
```

```
> at(female=1 white=1 year=3 ed=3 age=60 income=150)
```

```
Expression: Pr(class), predict(outcome())
```

lower	working	middle	upper	female	white	educ	age	income
0.230	0.634	0.133	0.003	0	0	1	25	30
0.008	0.138	0.759	0.095	1	1	3	60	150

```
Specified values of covariates
```

	year
Current	3

The differences between the ideal types are striking: While our first type has a predicted probability of only 0.13 of identifying as middle class, our second has a probability of 0.76. The second type has a probability of less than 0.01 of identifying as lower class, while the first type has a probability of 0.23. The example makes plain the large effect that these variables together have on class identification.

Although having the results in a single table is much more convenient than having to combine results from four *margins* commands, we also did several things to make the output clearer. First, we used value labels for the dependent variable *class* to label the columns with predictions. Because it is easy to be confused about the outcome categories when using these models, we advise always assigning clear value labels to your dependent variable (see chapter 2). Option *atright* places the values of the covariates to the right of the predictions. Because the values of the covariates clearly identify the rows, we turned off the row numbers in the table of predictions by using *norownum*. And to fit the results more compactly, we specified the column widths with *width(7)*.

7.12.1 (Advanced) Testing differences between ideal types

Although we regard this section as extremely useful, we mark it as advanced because it requires a firm grasp of using loops and local macros in Stata. If you are still getting used to these, you might want to skip this section until you are comfortable with both.

We may want to know whether a difference between ideal types is statistically significant for the same reason that we may perform a significance test for a set of coefficients: we are considering a change that involves multiple variables, and we want to evaluate how likely it is that we would observe a difference this large just by chance. Although the differences between predictions for our two ideal types are almost certainly significant, we can test this by extending methods used for binary outcomes.

To test differences in predictions, we need to overwrite the estimation results from `ologit` with the predictions generated by `margins`. An inherent limitation in `margins` is that posting can only be done for a single outcome. That is, we cannot post the predictions for our four outcomes at one time. (We hope this will be addressed in future versions of `margins`.) To deal with this inconvenience, we will use a `forvalues` loop to repeat the tests for each outcome. First, we fit the model and store the estimates, because we will have to restore the model results after we post the predictions for a particular outcome:

```
. ologit class i.female i.white i.year i.educ c.age##c.age income, nolog
      (output omitted)
. estimates store olm
```

Next, we compute tests for each of the four outcome categories by using the following commands:

```
. mlincom, clear
. forvalues iout = 1/4 { // start loop
2.     quietly {
3.         mtable, out(`iout') post
>         at(female=0 white=0 year=3 ed=1 age=25 income=30)
>         at(female=1 white=1 year=3 ed=3 age=60 income=150)
4.         mlincom 1 - 2, stats(est pvalue) rowname(outcome `iout') add
5.         estimates restore olm
6.     }
7. } // end loop
```

We start with `mlincom, clear` to erase previous results from `mlincom` before we accumulate the new results that we will use to make our table. The `forvalues { ... }` loop runs the code between braces once for each outcome. We use `quietly { ... }` so that the output from `mtable` and `mlincom` is not displayed. Instead, we will list results after the loop is completed. Within the loop, the `mtable` option `post` saves the estimates

for outcome `1` to the matrix `e(b)` so that `mlincom` can test the difference between the predictions for the first and second ideal types. The `mlincom` command uses option `add` to collect the results to display later.

After the loop, running `mlincom` without options displays the results we just computed. The column named `lincom` has the linear combination of the estimates—in this case, the difference in predictions—while column `pvalue` is the p -value for testing that the difference is 0:

```
. mlincom
```

	lincom	pvalue
outcome 1	0.222	0.000
outcome 2	0.496	0.000
outcome 3	-0.626	0.000
outcome 4	-0.092	0.000

As we anticipated, the predicted probabilities are significantly different for the two ideal types for each of the four outcomes.

7.13 Tables of predicted probabilities

When there are substantively important categorical predictors in the model, examining tables of predicted probabilities over values of these variables can be an effective way to interpret the results. In this example, we use `mtable` to look at predictions over the values of the year of the survey, which correspond to 1980, 1996, and 2012:

```
. mtable, at(year=(1 2 3)) atmeans norownum
Expression: Pr(class), predict(outcome())
```

year	lower	working	middle	upper
1	0.049	0.473	0.462	0.016
2	0.053	0.489	0.443	0.015
3	0.078	0.565	0.347	0.010

Specified values of covariates

	1. female	1. white	2. educ	3. educ	age	income
Current	.549	.814	.582	.241	45.2	68.1

The `atmeans` option holds other variables at their means in the estimation sample. We conclude the following:

Changing only the year of the survey, and with income measured in 2012 dollars for all survey years, the probability of a respondent identifying as working class increased from 0.47 in 1980 to 0.57 in 2012, while the probability of identifying as middle class declined from 0.46 to 0.35.

To obtain confidence intervals for the predictions, we use the option `stat(ci)`, which could be abbreviated simply as `ci`:

```
. mtable, at(year=(1 2 3)) atmeans stat(ci)
```

```
Expression: Pr(class), predict(outcome())
```

	year	lower	working	middle	upper
Pr(y)	1	0.049	0.473	0.462	0.016
ll	1	0.042	0.447	0.433	0.013
ul	1	0.056	0.499	0.491	0.020
Pr(y)	2	0.053	0.489	0.443	0.015
ll	2	0.046	0.468	0.421	0.012
ul	2	0.059	0.510	0.466	0.018
Pr(y)	3	0.078	0.565	0.347	0.010
ll	3	0.068	0.543	0.321	0.008
ul	3	0.088	0.587	0.372	0.012

Specified values of covariates

	1. female	1. white	2. educ	3. educ	age	income
Current	.549	.814	.582	.241	45.2	68.1

The lower and upper bounds of the intervals print on separate rows beneath each prediction. For example:

Holding independent variables at their sample means, respondents in 2012 had a 0.078 probability of identifying as lower class (95% CI: [0.068, 0.088]).

We might also want to generate tables for a combination of categorical independent variables. For example, how does class affiliation vary by race for the three years of our survey?⁶

```
. mtable, at(year=(1 2 3) white=(0 1)) atmeans norownum
```

```
Expression: Pr(class), predict(outcome())
```

white	year	lower	working	middle	upper
0	1	0.059	0.511	0.417	0.013
0	2	0.063	0.526	0.399	0.012
0	3	0.093	0.593	0.306	0.008
1	1	0.047	0.464	0.472	0.017
1	2	0.051	0.480	0.454	0.016
1	3	0.075	0.558	0.356	0.010

Specified values of covariates

	1. female	2. educ	3. educ	age	income
Current	.549	.582	.241	45.2	68.1

6. While we are considering the probabilities implied by having `year` and `white` in the model as separate independent variables, we could also fit a model in which the interaction term `i.year#i.white` is included.

The predictions vary by year within a given value of `white`. The way in which variables vary is determined by the order in which the variables are specified with `ologit`, not by the order of variables within the `at()` statement. The table might be clearer if predictions were arranged to vary by `white` within each value of `year` (in practice, we often have to try it both ways before deciding which is clearer for the purpose at hand). We can refit the `ologit` model with `year` listed before `white`, or we can specify the values of `year` within three separate `at()` statements:

```
. mtable, atmeans norovnum
>   at(year=1 white=(0 1)) // 1980
>   at(year=2 white=(0 1)) // 1996
>   at(year=3 white=(0 1)) // 2012
```

Expression: Pr(class), predict(outcome())

white	year	lower	working	middle	upper
0	1	0.059	0.511	0.417	0.013
1	1	0.047	0.464	0.472	0.017
0	2	0.063	0.526	0.399	0.012
1	2	0.051	0.480	0.454	0.016
0	3	0.093	0.593	0.306	0.008
1	3	0.075	0.558	0.356	0.010

Specified values of covariates

	1. female	2. educ	3. educ	age	income
Current	.549	.582	.241	45.2	68.1

The results are exactly the same as before with only the rows rearranged.

In these results, the specified values of the covariates are the same for all predictions, namely, their global means. A different possibility is that we want the values of other variables to vary depending on a person's race and the year of the survey. For example, suppose that we want to compare whites and nonwhites for different survey years, holding all other variables constant at their local means within each race-year group. In other words, instead of looking at predictions when all independent variables except race and year are held to the same values, we compute predictions when the values of the other independent variables vary according to the means for whites and for blacks in different years. To do this, we specify the `over()` option:


```
. mtable, over(year white) atmeans
Expression: Pr(class), predict(outcome())
```

	1. female	white	year	2. educ	3. educ	age
1	.659	0	1	.447	.136	42.6
2	.539	1	1	.553	.166	44.5
3	.617	0	2	.597	.199	40.1
4	.534	1	2	.609	.261	44.7
5	.576	0	3	.56	.234	44.1
6	.538	1	3	.581	.31	48.9
	income	lower	working	middle	upper	
1	46.5	0.093	0.592	0.307	0.008	
2	67.7	0.054	0.493	0.439	0.015	
3	50.5	0.085	0.579	0.327	0.009	
4	70.5	0.048	0.467	0.468	0.017	
5	54.3	0.111	0.615	0.266	0.007	
6	77.8	0.058	0.507	0.422	0.014	

Specified values where .n indicates no values specified with at()

	No at()
Current	.n

The means of the independent variables are included in the table and are different for each row. For example, the values of *female*, *white*, *year*, *educ*, *age*, and *income* in row 1 are the means for the subsample that is black and responded to the survey in 1980:

```
. sum female i.educ age income if white==0 & year==1, sep(9)
```

Variable	Obs	Mean	Std. Dev.	Min	Max
female	132	.6590909	.4758206	0	1
educ					
hs only	132	.4469697	.4990739	0	1
college	132	.1363636	.3444816	0	1
age	132	42.55303	16.8257	18	83
income	132	46.48708	55.15685	1.57795	267.8147

Returning to the output from *mtable*, if we look at variable 3. *educ*, for example, the difference between row 1 and row 2 shows that a higher proportion of white respondents had college degrees in 1980 (0.166) than did black respondents (0.136). The differences between rows 1 and 3 and between rows 2 and 4, on the other hand, reflect that the proportion of respondents with a college degree increased between 1980 and 1996.

When there are many regressors in the table, it might be easier to see the changing predictions by listing the value of the regressors (that is, the *at()* variables) on the right by using the *atright* option:


```
. mtable, over(year white) atmeans atright
Expression: Pr(class), predict(outcome())
```

		lower	working	middle	upper	1. female	white
1		0.093	0.592	0.307	0.008	.659	0
2		0.054	0.493	0.439	0.015	.539	1
3		0.085	0.579	0.327	0.009	.617	0
4		0.048	0.467	0.468	0.017	.534	1
5		0.111	0.615	0.266	0.007	.576	0
6		0.058	0.507	0.422	0.014	.538	1
		2. year	3. educ	educ	age	income	
1		1	.447	.136	42.6	46.5	
2		1	.553	.166	44.5	67.7	
3		2	.597	.199	40.1	50.5	
4		2	.609	.261	44.7	70.5	
5		3	.56	.234	44.1	54.3	
6		3	.581	.31	48.9	77.8	

Specified values where .n indicates no values specified with at()

	No at()
Current	.n

Using `over()` in this way yields predictions that reflect the effects of both race and year and also compositional differences over race and year in the means of the other variables. In this respect, the predictions are not as simple to interpret as the examples above in which the values of other characteristics were the same regardless of year and race. Comparing the predictions we just made with those we made holding the independent variables to the same values for everyone, we can see the probabilities vary more when the means vary over groups. Substantively, this indicates that the differences in the population composition by race and year result in differences in predictions that are larger than they would be if we assumed that population characteristics were constant across race and time.

7.14 Plotting predicted probabilities

Plotting predicted probabilities for each outcome can also be useful for the ORM. These plots illustrate how predicted probabilities change as a continuous independent variable changes. With the BRM, we showed two approaches for making plots: directly with `marginsplot` or in two steps with `mgen` and `graph`. Plotting multiple outcomes, however, can only be done using the latter technique because `marginsplot` is limited to plotting a single outcome.

To illustrate graphing predictions, we consider how the probability of class affiliation changes as household income changes, holding all other variables at their sample means. Of course, the plot could also be constructed for other sets of characteristics. The option `at(inc=0(25)250)` tells `mgen` to generate predictions as `income` changes from 0 to 250

in increments of 25, leading to 11 sets of predictions. The option `atmeans` holds other variables to their means. We use `stub(CL_)` to add `CL_` (indicating predictions for class) to the names of variables generated by `mgen`:

```
. mgen, at(income=(0(25)250)) stub(CL_) atmeans
Predictions from: margins, at(income=(0(25)250)) atmeans predict(outcome())
```

Variable	Obs	Unique	Mean	Min	Max	Label
CL_pr1	11	11	.0439624	.0074601	.1207294	pr(y=lower) from margins
CL_ll1	11	11	.0385464	.0057745	.1067216	95% lower limit
CL_ul1	11	11	.0493784	.0091458	.1347373	95% upper limit
CL_income	11	11	125	0	250	household income
CL_Cpr1	11	11	.0439624	.0074601	.1207294	pr(y<=lower)
CL_pr2	11	11	.377087	.1301718	.6238775	pr(y=working) from margins
CL_ll2	11	11	.3565982	.1081524	.6070259	95% lower limit
CL_ul2	11	11	.3975758	.1521912	.6407292	95% upper limit
CL_Cpr2	11	11	.4210494	.137632	.744607	pr(y<=working)
CL_pr3	11	11	.5424338	.2492696	.7612023	pr(y=middle) from margins
CL_ll3	11	11	.5212693	.2296227	.7417022	95% lower limit
CL_ul3	11	11	.5635983	.2689165	.7807024	95% upper limit
CL_Cpr3	11	11	.9634833	.8988342	.9938766	pr(y<=middle)
CL_pr4	11	11	.0365167	.0061234	.1011657	pr(y=upper) from margins
CL_ll4	11	11	.030147	.0047639	.0828273	95% lower limit
CL_ul4	11	11	.0428865	.0074829	.1195042	95% upper limit
CL_Cpr4	11	1	1	1	1	pr(y<=upper)

Specified values of covariates

1.	1.	2.	3.	2.	3.	
female	white	year	year	educ	educ	age
.5491103	.8140569	.4510676	.3099644	.5818505	.2414591	45.15712

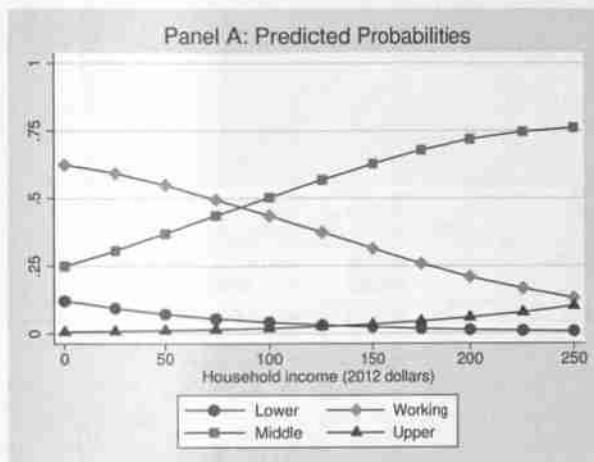
Each variable has 11 observations corresponding to different values of `income`. Variables containing predicted probabilities are stored in variables named `CL_pr#`. For example, `CL_pr2` is the predicted probability of identifying as working class, the second category of our outcome. Variables containing cumulative probabilities—that is, the probability of observing a given category or lower—are stored as variables `CL_Cpr#`. For example, `CL_Cpr2` is the predicted probability of a respondent identifying as either lower class or working class.

Although `mgen` assigns variable labels to the variables it generates, we can change these to improve the look of the plot that we are creating. Specifically, we use

```
. label var CL_pr1 "Lower"
. label var CL_pr2 "Working"
. label var CL_pr3 "Middle"
. label var CL_pr4 "Upper"
. label var CL_Cpr1 "Lower"
. label var CL_Cpr2 "Lower/Working"
. label var CL_Cpr3 "Lower/Working/Middle"
```


Next, we plot the probabilities of individual outcomes by using `graph`. Here we plot the four probabilities against values of income.

```
. graph twoway connected CL_pr1 CL_pr2 CL_pr3 CL_pr4 CL_income,
>   title("Panel A: Predicted Probabilities")
>   xtitle("Household income (2012 dollars)")
>   xlabel(0(50)250) ylabel(0(.25)1, grid gmin gmax)
>   ytitle("") name(tmpprob, replace)
```



Standard options for `graph` are used to specify the axes and labels. The `name(tmpprob, replace)` option saves the graph with the name `tmpprob` so that we can combine it with our next graph, which plots the cumulative probabilities. The values on the y axis of each of the four lines indicate the predicted probabilities of each category when income equals the value on the x axis and other variables are held at their means. At all values of income, these probabilities sum to 1. We will wait to discuss what it is showing substantively until after we complete the graph with cumulative probabilities.

A graph of cumulative probabilities uses lines to indicate the probability that $y \leq \#$ rather than $y = \#$. To create this graph, we use the command

```
. graph twoway connected CL_Cpr1 CL_Cpr2 CL_Cpr3 CL_income,
>   title("Panel B: Cumulative Probabilities")
>   xtitle("Household income (2012 dollars)")
>   xlabel(0(50)250) ylabel(0(.25)1, grid gmin gmax)
>   ytitle("") name(tmpcprob, replace)
```

which saves the resulting graph as `tmpcprob`. Next, we combine these two graphs (see chapter 2 for details on combining graphs):

```
. graph combine tmpprob tmpcprob, col(1) iscale(*.9) imargin(small)
>   ysize(4.6) xsize(3.287) caption("Other variables held at their means")
```


This leads to figure 7.2. Panel A plots the predicted probabilities for each outcome and shows that the probabilities of working class and middle class are larger than the probabilities of lower class and upper class. As income increases, the probability of a respondent identifying as lower class or working class decreases, while the probability of identifying as middle class or upper class increases. Panel B plots the cumulative probabilities. Both panels present the same information. In applications, you should use the graph that you find most effective.

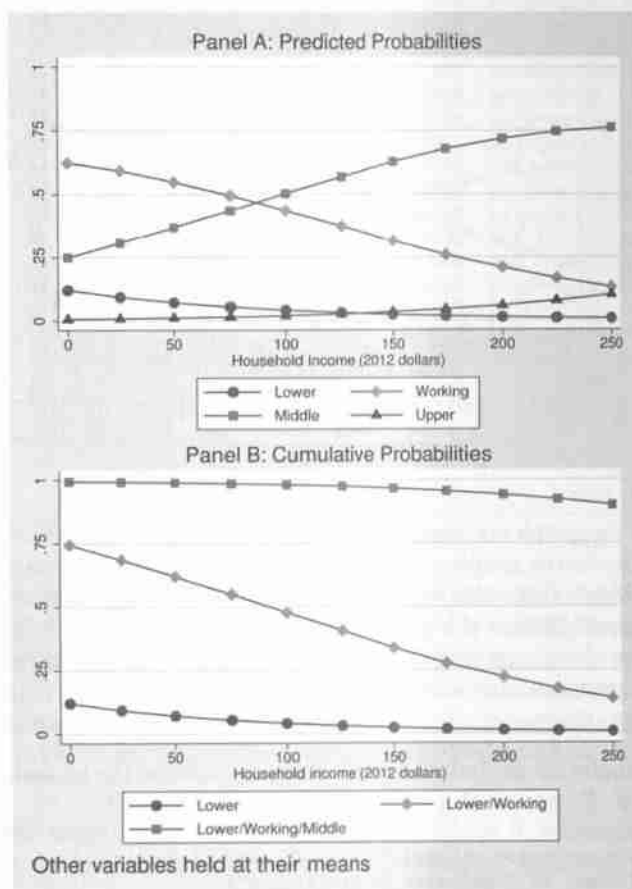
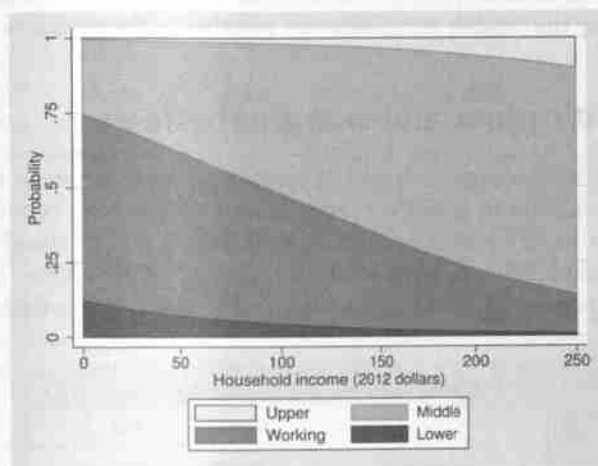


Figure 7.2. Plot of predicted probabilities and cumulative probabilities for the ordered logit model

A different way of plotting cumulative probabilities is to use shading instead of lines, which looks like this:



To create this graph, we use a `twoway area` plot type (see [G-2] `twoway area`). In our prior graph, `CL_Cpr#` designated the height on the *y* axis at which a line should be drawn. In a `twoway area` plot, the value of `CL_Cpr#` indicates that the graph should be shaded from that value to the bottom of the graph. We used the following commands:

```
. capture drop one
. gen one = 1
. label variable one "Upper"
. label variable CL_Cpr3 "Middle"
. label variable CL_Cpr2 "Working"
. label variable CL_Cpr1 "Lower"
. graph twoway
>   (area one CL_income, fcolor(gs15))
>   (area CL_Cpr3 CL_income, fcolor(gs11))
>   (area CL_Cpr2 CL_income, fcolor(gs7))
>   (area CL_Cpr1 CL_income, fcolor(gs3)),
>   xtitle("Household income (2012 dollars)") ytitle("Probability")
>   xlabel(0(50)250) ylabel(0(.25)1, grid gmax)
```

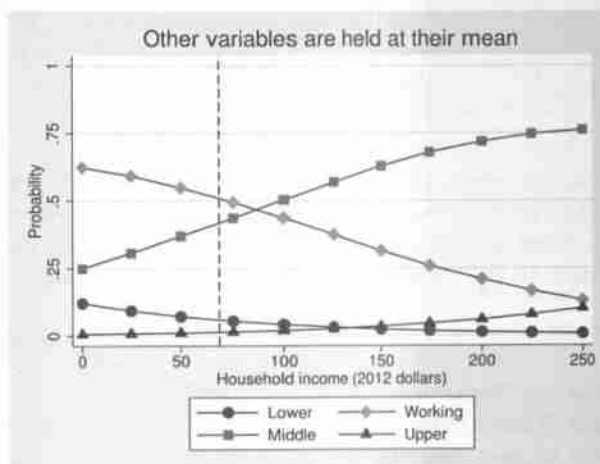
First, we generate the variable `one`, whose values are all 1. An `area` plot using this variable is simply a solid block from the top of the graph to the bottom, which we use to depict the probability of the highest category ("Upper"). Accordingly, we label this variable with the name of highest category. Next, we label the variables for the other cumulative probabilities according to the highest category each represents. Then, using the `graph twoway` command, we draw four `area` plots, each on top of the other to make a single graph. The first plot covers the entire plot region (everything below the value of variable `one`, which is always equal to 1), which we shade using `fcolor(gs15)`.⁷ The

7. The `fcolor()` option uses `gs#` as names of grayscale colors, and `gs15` is the lightest.

other area plots use the three cumulative probability variables and are shaded using progressively darker grays. The shading allows us to easily see how identification as middle class expands as household income increases. We can also quickly see how large the probabilities for the middle categories are relative to the extremes.

7.15 Probability plots and marginal effects

Having considered various methods of interpretation, we now show the link between marginal effects and plots of predicted probabilities to hopefully provide you with new insights on the nature of ordinal models. The following graph, based on section 7.14, shows how the probabilities of class affiliation change with income, holding all other variables at their means. The mean of income is indicated with a dashed, vertical line:



The slope of each probability curve evaluated at the mean of income, indicated by where the probability curves intersect the vertical line, is the marginal change in the probability of a given class affiliation with respect to income, with all variables held at their means. We can compute these changes by using `mchange`, `atmeans` to estimate MEMS:


```
. mchange income, atmeans amount(marginal) dec(4)
ologit: Changes in Pr(y) | Number of obs = 5620
Expression: Pr(class), predict(outcome())
```

	lower	working	middle	upper		
income						
Marginal	-0.0006	-0.0022	0.0027	0.0002		
p-value	0.0000	0.0000	0.0000	0.0000		
Predictions at base value						
	lower	working	middle	upper		
Pr(y base)	0.0586	0.5107	0.4173	0.0134		
Base values of regressors						
	1. female	1. white	2. year	3. year	2. educ	3. educ
at	.5491	.8141	.4511	.31	.5819	.2415
	age	income				
at	45.16	68.08				

1: Estimates with margins option atmeans.

The marginal changes are in row **Marginal**, with the significance level for the test that the change is 0 listed in row **p-value**. These changes correspond to the slopes of the probability curves at the point of intersection with the vertical line. For example, the slope for middle class, shown with squares, is 0.0027.

The magnitude of the marginal changes would differ if we computed the marginal effects at different values of the independent variables. For example, we can compute the effects with income equal to \$250,000, with all other variables still kept at their means:


```
. mchange income, at(income=250) atmeans amount(marginal) dec(4)
ologit: Changes in Pr(y) | Number of obs = 5620
Expression: Pr(class), predict(outcome())
```

	lower	working	middle	upper		
income						
Marginal	-0.0001	-0.0013	0.0003	0.0011		
p-value	0.0000	0.0000	0.0378	0.0000		
Predictions at base value						
	lower	working	middle	upper		
Pr(y base)	0.0075	0.1302	0.7612	0.1012		
Base values of regressors						
	1. female	1. white	2. year	3. year	2. educ	3. educ
at	.5491	.8141	.4511	.31	.5819	.2415
	age	income				
at	45.16	250				

1: Estimates with margins option atmeans.

The marginal change for the probability of identifying with the middle class is much smaller, corresponding to the leveling off of the curve (shown with ■'s) on the right side of the graph.

In this example, the signs of the marginal effects for each outcome are the same throughout the range of income. This, however, does not need to be true. In the ORM, not only does the magnitude of the effect change as the values of the independent variables change, but even the sign can change. That is to say, the effect of a variable can be positive at one point and can be negative at other points, even if we have not included polynomial terms or interactions in the model. In a model without interaction or polynomials for a given independent variable, the sign of that variable's regression coefficient will always be the same as the direction of changes in the probability of the highest outcome category as the independent variable increases.

In our example of subjective social class, because the coefficient for income is positive, increases in income will always increase the probability of identifying as upper class.⁸ Conversely, the change in the probability of the lowest category will be in the opposite direction as the regression coefficient. Thus increases in income always decrease the probability of identifying as lower class. The middle categories are more complicated. In terms of the latent variable model described in section 7.1.1, as income increases, some people shift from lower class to working class, while other people shift from working class to middle class. The specific implication for the probability of identifying as working

8. Of course, there is no reason this must be true substantively. For example, one might hypothesize that income increases the predicted probability of identifying as upper class only up to a point, after which there is no effect. Modeling this would involve adding additional terms for the effect of income to the model, for example, by using splines or polynomials.

class depends on whether the proportion entering the category from below is bigger or smaller than the proportion exiting from above. As a result, the sign of the predicted change in middle categories can change when computed at different values of income. Our running example does not provide a good illustration of this (for reasons explained below), so we use an example from chapter 8. Party affiliation is coded as strong Democrat; Democrat, Independent, or Republican; and strong Republican. We fit an OLM using age, income, race, gender, and education as predictors:

```
. use partyid4, clear
(partyid4.dta | 1992 American National Election Study | 2014-03-12)
. ologit partystrong age income i.black i.female i.educ, nolog
Ordered logistic regression      Number of obs   =    1382
                                LR chi2(6)        =    173.85
                                Prob > chi2       =    0.0000
                                Pseudo R2        =    0.0755
Log likelihood = -1064.4742
```

partystrong	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
age	-.0081028	.0036953	-2.19	0.028	-.0153456	-.0008601
income	.0090361	.002415	3.74	0.000	.0043028	.0137694

(output omitted)

Notice that the coefficient for income is positive. Next, we compute predictions as income increases from \$0 to \$200,000:

```
. mgen, atmeans at(income=(0(20)200)) stub(olm) replace
Predictions from: margins, atmeans at(income=(0(20)200)) predict(outcome())
Variable   Obs Unique   Mean   Min   Max   Label
olmpr1     11     11   .1148384 .0444239 .2207461 pr(y=StrDem) from margins
olml1      11     11   .0838875 .0106582 .1826952 95% lower limit
olmul1     11     11   .1457892 .0781895 .258797 95% upper limit
olmincome  11     11     100      0     200 Income in $1,000s
```

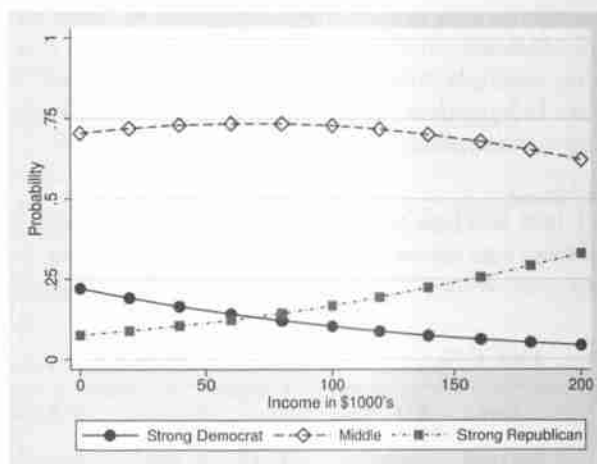
(output omitted)

Specified values of covariates

	1.	1.	2.	3.
age	black	female	educ	educ
45.94645	.1374819	.4934877	.5803184	.2590449

```
. label var olmpr1 "Strong Democrat"
. label var olmpr2 "Middle"
. label var olmpr3 "Strong Republican"
```


Plotting these predictions produces the following graph:



As income increases, the probability of being a strong Republican increases steadily, while the probability of being a strong Democrat decreases. The changes in the probability of being “in the middle” are more complex. As income increases from \$0 to about \$50,000, the probability increases, and then it decreases till \$200,000. That is, the marginal effect of income on being politically in the middle is both positive and negative depending on where it is evaluated. We can see this by using `mchange` evaluated at the levels of income:

```
. * discrete change at income of $0
. mchange income, at(income=0) atmeans amount(sd) stat(change) brief
ologit: Changes in Pr(y) | Number of obs = 1382
Expression: Pr(partystrong), predict(outcome())
```

		StrDem	Middle	StrRep
income	+SD	-0.040	0.021	0.019

```
. * discrete change at income of $60,000
. mchange income, at(income=68) atmeans amount(sd) stat(change) brief
ologit: Changes in Pr(y) | Number of obs = 1382
Expression: Pr(partystrong), predict(outcome())
```

		StrDem	Middle	StrRep
income	+SD	-0.026	-0.005	0.031


```

. * discrete change at income of $200,000
. mchange income, at(income=200) atmeans amount(sd) stat(change) brief
ologit: Changes in Pr(y) | Number of obs = 1382
Expression: Pr(partystrong), predict(outcome())

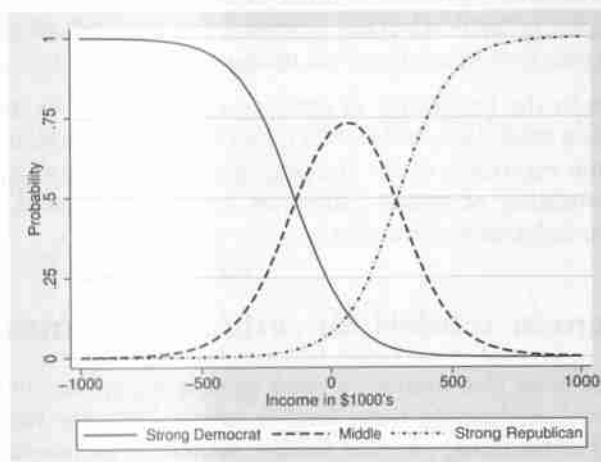
```

	StrDem	Middle	StrRep
income			
+SD	-0.010	-0.048	0.058

The changes in the probability of outcome 1, being a strong Democrat, are always negative and get smaller as the probability approaches 0. Changes in the probability of outcome 3, being a strong Republican, steadily increase as income increases. For the middle category, the change is positive, then 0, and then negative.

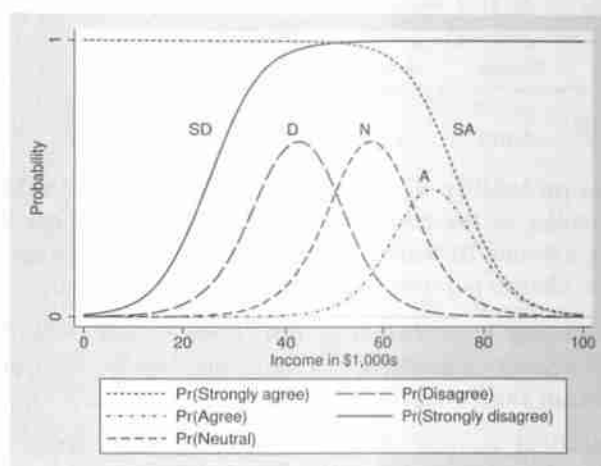
This pattern of change in probabilities must hold for any ORM. Indeed, Anderson (1984) made this a defining characteristic of an ordinal model; see Long (Forthcoming) for further discussion of this property.

When an independent variable changes over an extended range (technically, from negative infinity to positive infinity) the plot of probabilities must have a pattern similar to the following graph that extends the range of income from our example above:



The height of the bell-shaped curve for the middle category depends on the distance between thresholds, which in turn depends on the relative size of the outcome categories. The observed range for the independent variable—income, in this case—might fall anywhere within this graph. For example, if our sample included only those cases with an income about \$68,000, corresponding to the peak of the probability curve for the middle category, our graph would show that the probability of being politically in the middle always decreases as income increases. Indeed, we had to change our example for this section because the upper- and lower-class categories in our example of subjective social class were so small.

If you have more categories, the probability curves will be ordered from left to right as illustrated in this graph:



This pattern of curves will also occur in other ordinal models. As shown in chapter 8, if this pattern does not correspond to the process being modeled, an ordinal model will force the data into this pattern and provide misleading results.

In sum, changes in the probability of extreme categories in the ORM are always in the opposite direction from one another; the direction of the change for either category will remain the same regardless of the starting value or the magnitude of the change. Changes in the probability of middle categories, on the other hand, can change sign over the range of an independent variable.

7.16 Less common models for ordinal outcomes

Stata can also fit several less commonly used models for ordinal outcomes. In concluding this chapter, we describe these models briefly and note their commands for estimation. Long (Forthcoming) provides further details. `SPost` commands do not work with all these models, but our `m*` commands do work with the estimation commands that support `margins`.

7.16.1 The stereotype logistic model

The stereotype logistic model, also referred to as the stereotype ORM, was proposed by Anderson (1984) in response to the restrictive assumption of parallel regressions in the ORM. The stereotype logistic model is a compromise between allowing the coefficients for each independent variable to vary by outcome category (as is the case with the multinomial logit model, considered in the next chapter) and restricting the coefficients

to be identical across all outcomes (as was the case with the ordered logit model). The stereotype logistic model can be fit in Stata by using the `slogit` command (see [R] `slogit`). The one-dimensional version of the model is defined as

$$\ln \frac{\Pr(y = q | \mathbf{x})}{\Pr(y = r | \mathbf{x})} = (\theta_q - \theta_r) - (\phi_q - \phi_r)(\mathbf{x}\beta)$$

where β is a vector of coefficients associated with the independent variables, the θ 's are intercepts, and the ϕ 's are scale factors that mediate the effects of the x 's. This one-dimensional model is ordinal as defined in section 7.15 and often produces very similar predictions to the ORM. When additional dimensions are added, it is no longer an ordinal model. Indeed, with enough dimensions, it is equivalent to the multinomial logit model. Accordingly, we postpone further discussion until chapter 8.

7.16.2 The generalized ordered logit model

The parallel regression assumption results from assuming the same coefficient vector β in the $J - 1$ logit equations

$$\ln \Omega_{\leq m | > m}(\mathbf{x}) = \tau_m - \mathbf{x}\beta$$

where $\Omega_{\leq m | > m}(\mathbf{x}) = \Pr(y \leq m | \mathbf{x}) / \Pr(y > m | \mathbf{x})$. The generalized ordered logit model allows β to differ for each of the $J - 1$ comparisons. That is,

$$\ln \Omega_{\leq m | > m}(\mathbf{x}) = \tau_m - \mathbf{x}\beta_m \quad \text{for } j = 1 \text{ to } J - 1$$

where predicted probabilities are computed as

$$\begin{aligned} \Pr(y = 1 | \mathbf{x}) &= \frac{\exp(\tau_1 - \mathbf{x}\beta_1)}{1 + \exp(\tau_1 - \mathbf{x}\beta_1)} \\ \Pr(y = j | \mathbf{x}) &= \frac{\exp(\tau_j - \mathbf{x}\beta_j)}{1 + \exp(\tau_j - \mathbf{x}\beta_j)} - \frac{\exp(\tau_{j-1} - \mathbf{x}\beta_{j-1})}{1 + \exp(\tau_{j-1} - \mathbf{x}\beta_{j-1})} \quad \text{for } j = 2 \text{ to } J - 1 \\ \Pr(y = J | \mathbf{x}) &= 1 - \frac{\exp(\tau_{J-1} - \mathbf{x}\beta_{J-1})}{1 + \exp(\tau_{J-1} - \mathbf{x}\beta_{J-1})} \end{aligned}$$

No formal constraint precludes negative predicted probabilities. Discussions of this model can be found in Clogg and Shihadeh (1994, 146–147), Fahrmeir and Tutz (1994, 91), and McCullagh and Nelder (1989, 155). A critical view of the model can be found in Greene and Hensher (2010, 189–192), who highlight that the model can predict negative “probabilities” and that it cannot be formulated in terms of a continuous latent dependent variable. Further, as noted by Long (Forthcoming), the generalized ordered logit model is not an ordinal regression model because, like the multinomial logit model, it does not necessarily make predictions that maintain the ordinality of the outcome.

The model can be fit in Stata with the `gologit2` command (Williams 2005). The command does not work with factor variables, so categorical variables in the model need to be constructed as a series of binary variables. Also, because we can no longer use factor-variable notation to include age-squared in the model, we need to create an age-squared variable explicitly:

```
. use gssclass4, clear
(gssclass4.dta | GSS Subjective Class Identification | 2013-11-20)
. gen year1996 = (year==2) if year < .
. gen year2012 = (year==3) if year < .
. gen educ_hs = (educ==2) if educ < .
. gen educ_col = (educ==3) if educ < .
. gen agesq = age*age if age < .
```

The specification of the dependent variable and independent variables is otherwise like the other estimation commands we consider. We estimate the parameters of the model by specifying the `or` option to obtain odds ratios:


```
. gologit2 class female white year1996 year2012 educ_hs educ_col
> age agesq income, nolog or
```

Generalized Ordered Logit Estimates

Number of obs	=	5620
LR chi2(27)	=	1782.34
Prob > chi2	=	0.0000
Pseudo R2	=	0.1552

Log likelihood = -4852.0145

class	Odds Ratio	Std. Err.	z	P> z	[95% Conf. Interval]	
lower						
female	.8709876	.0995261	-1.21	0.227	.6962205	1.089625
white	1.11671	.1404256	0.88	0.380	.8727749	1.428823
year1996	.8572836	.1356375	-0.97	0.330	.6287085	1.16896
year2012	.4740378	.074985	-4.72	0.000	.3476697	.6463373
educ_hs	1.395278	.1800471	2.58	0.010	1.083482	1.796802
educ_col	4.315005	1.130562	5.58	0.000	2.582025	7.211111
age	.9300842	.0159959	-4.21	0.000	.8992553	.96197
agesq	1.000774	.0001691	4.58	0.000	1.000443	1.001106
income	1.052365	.0036226	14.83	0.000	1.045289	1.059489
_cons	9.397178	4.093324	5.14	0.000	4.001492	22.06851
working						
female	1.0751	.064515	1.21	0.228	.9558058	1.209283
white	1.30927	.1050703	3.36	0.001	1.118715	1.532284
year1996	.9415489	.0704209	-0.81	0.421	.8131661	1.090201
year2012	.7055537	.0593443	-4.15	0.000	.5983224	.832003
educ_hs	1.329956	.1141584	3.32	0.001	1.124018	1.573625
educ_col	4.742383	.5034121	14.66	0.000	3.85159	5.839197
age	.9538031	.0096077	-4.70	0.000	.9351571	.972821
agesq	1.000728	.0001015	7.18	0.000	1.000529	1.000927
income	1.011296	.000668	17.00	0.000	1.009987	1.012606
_cons	.3530586	.0870504	-4.22	0.000	.2177579	.5724264
middle						
female	1.015499	.1568278	0.10	0.921	.7502826	1.374467
white	.7961042	.175904	-1.03	0.302	.5162878	1.227575
year1996	.9762229	.1919355	-0.12	0.903	.6640396	1.435172
year2012	.4899366	.1130594	-3.09	0.002	.3116834	.7701334
educ_hs	.6469348	.1694775	-1.66	0.096	.3871428	1.08106
educ_col	1.779687	.4778177	2.15	0.032	1.051501	3.012158
age	.9865854	.0272608	-0.49	0.625	.9345763	1.041489
agesq	1.00033	.0002667	1.24	0.216	.9998074	1.000853
income	1.011058	.0009115	12.20	0.000	1.009273	1.012846
_cons	.0139257	.0097749	-6.09	0.000	.0035183	.055119

We have three sets of odds ratios labeled *lower*, *working*, and *middle*, compared with one set when we used *ologit*. The odds ratios indicate the factor change in odds of observing a value above the listed category versus observing values at or below the listed category. Accordingly, we can interpret the three coefficients for *white* as follows:

Holding all other variables constant, white respondents have 1.12 times higher odds of identifying themselves as working, middle, or upper class than do nonwhite respondents. White respondents have 1.31 times higher odds of identifying themselves as middle or upper class than do nonwhite respondents. And, white respondents have 0.80 times lower odds of identifying themselves as upper class than do nonwhite respondents.

The key difference between the generalized and the ordered logit models is that in the ordered logit model, the odds ratios described in these three sentences are constrained to be equal. Based on the ordered logit model, we would replace 1.12, 1.31, and 0.80 in the paragraph above with the same value 1.27.

`gologit2` allows users to fit the model with some of the coefficients constrained to be equal, as in `ologit`, while others are allowed to vary. In addition to this, `gologit2` can fit two special cases of the general model: the proportional-odds model and the partial proportional-odds model (Lall, Walters, and Morgan 2002; Peterson and Harrell 1990). These models are less restrictive than the ordinal logit model fit by `ologit`, but they are more parsimonious than the multinomial logit model fit by `mlogit`.

7.16.3 (Advanced) Predictions without using factor-variable notation

Factor variables make it much simpler to make predictions when there are linked variables, such as `age` and `age-squared`. Because `gologit2` does not support factor-variable notation, we use this model to illustrate how to make the correct predictions. The most important point for most readers is likely that you want to use factor variables whenever possible! If you use factor-variable notation in your models, you do not need to worry about the issues discussed in this section. However, you might still find the section useful to deepen your understanding of predictions in nonlinear models.

We can compute predicted probabilities for given values of observations as we did with the ordered logit model and use the same approach to interpretation. For example, here are results using the same ideal types that we used in section 7.12.

```
mtable, atright norownum width(7) ///
  at(female=0 white=0 year1996=0 year2012=1 educ_hs=0 educ_col=0 ///
    age=25 agesq=625 income=30) ///
  at(female=1 white=1 year1996=0 year2012=1 educ_hs=0 educ_col=1 ///
    age=60 agesq=3600 income=150)
```

Because `gologit2` does not support factor-variable notation, we must explicitly specify the values of indicator variables for educational degree and survey year. We also must specify that the value of `agesq` is the square of the value of `age`. As a consequence, the output is messier (though the predictions are correct):

Expression: `Pr(class), predict(outcome())`

lower	working	middle	upper	female	white	educ_col	age
0.155	0.701	0.136	0.008	0	0	0	25
0.000	0.122	0.809	0.069	1	1	1	60
agesq	income						
625	30						
3600	150						

Specified values of covariates

	year1996	year2012	educ_hs
Current	0	1	0

Comparing the results from `ologit` that were computed earlier in the chapter,

Expression: `Pr(class), predict(outcome())`

lower	working	middle	upper	female	white	educ	age	income
0.230	0.634	0.133	0.003	0	0	1	25	30
0.008	0.138	0.759	0.095	1	1	3	60	150

Specified values of covariates

	year
Current	3

the main difference between the generalized and the ordered logit models is that the predicted probabilities for the categories with the highest probabilities (working class for the first ideal type and middle class for the second) are about 0.06 higher in the generalized ordered logit model.

We can also use `mchange` to obtain changes in the predicted probability for particular values of the independent variables, which provides an opportunity to illustrate how to deal with polynomial terms, such as age-squared, when you are not using factor-variable notation. Suppose that we want the discrete change for `white`, which is a binary variable. If factor-variable notation had been used, `mchange` would know that it is a binary variable. Because we are not using factor-variable notation, we must tell `mchange` to compute the change from 0 to 1 with the option `amount(binary)`. It is tempting, but *incorrect*, to compute the change like this:

```
. mchange white, amount(binary) atmeans // incorrect method!
```

```
gologit2: Changes in Pr(y) | Number of obs = 5620
```

Expression: `Pr(class), predict(outcome())`

	lower	working	middle	upper
white				
0 to 1	-0.001	-0.066	0.072	-0.005
p-value	0.403	0.001	0.000	0.336
Predictions at base value				
	lower	working	middle	upper
Pr(y base)	0.011	0.503	0.466	0.020

Base values of regressors

	female	white	year1996	year2012	educ_hs	educ_col
at	.549	.814	.451	.31	.582	.241
	age	agesq	income			
at	45.2	2325	68.1			

1: Estimates with margins option atmeans.

Because the mean of `age` is 45.16, `agesq` should be held at $45.16 \times 45.16 = 2039$, not 2325, which is the mean of `agesq`.

The correct way to compute marginal effects is to specify the value of `agesq` in `at()`:

```
. mchange white, amount(binary) at(agesq=2039) atmeans
```

```
gologit2: Changes in Pr(y) | Number of obs = 5620
```

```
Expression: Pr(class), predict(outcome())
```

	lower	working	middle	upper
white				
0 to 1	-0.002	-0.064	0.070	-0.004
p-value	0.402	0.001	0.000	0.335

Predictions at base value

	lower	working	middle	upper
Pr(y base)	0.014	0.552	0.416	0.018

Base values of regressors

	female	white	year1996	year2012	educ_hs	educ_col
at	.549	.814	.451	.31	.582	.241
	age	agesq	income			
at	45.2	2039	68.1			

1: Estimates with margins option atmeans.

In this example, the differences are slight. Depending on the magnitudes of the coefficients for the linked variables and the distribution of those variables, however, the differences can be substantial. Fortunately, some simple programming tools can automate the process:

```
summarize age
local magesq = r(mean)*r(mean)
mchange white, amount(binary) at(agesq=`magesq`) atmeans
```

`summarize` computes the mean of `age`, which is returned in `r(mean)`. The local macro `magesq` is set equal to the mean times the mean. Within the `at()` specification, `agesq` is set equal to this local macro.

Another limitation caused by the lack of support for factor-variable notation in `gologit2` is that you cannot use `mchange` to compute marginal effects of `age` because `margins` has no way to know that `age` and `agesq` must change together. The only

solution is to compute the appropriate predictions, specifying both the value of age and age-squared at two values of age and then subtracting the predictions.

For categorical independent variables with more than two variables, we need to explicitly constrain the other indicator variables to 0 as we let one of the indicators change from 0 to 1. We can do this with `at()`. First, we compute the change between 1996 and 1980 (the base category), and then we compute the change between 2012 and 1980:

```
. * change from 1980 to 1996
. mchange year1996, amount(binary) at(year2012=0 agesq=2039) atmeans brief
logit2: Changes in Pr(y) | Number of obs = 5620
Expression: Pr(class), predict(outcome())
```

	lower	working	middle	upper
year1996				
0 to 1	0.002	0.013	-0.014	-0.001
p-value	0.325	0.469	0.436	0.903

```
. * change from 1980 to 2012
. mchange year2012, amount(binary) at(year1996=0 agesq=2039) atmeans brief
logit2: Changes in Pr(y) | Number of obs = 5620
Expression: Pr(class), predict(outcome())
```

	lower	working	middle	upper
year2012				
0 to 1	0.011	0.074	-0.073	-0.012
p-value	0.000	0.000	0.000	0.004

To compute the change from 1996 to 2012, we cannot use `mchange` because we need to change the value of two variables at once, namely, `year1996` and `year2012`. To estimate the changes, we use a simple program:

```
. estimates store go1m
. mlincom, clear
. forvalues iout = 1/4 {
2.     qui {
3.         mtable, atmeans post outcome(`iout')
>         at(year1996=1 year2012=0 agesq=2039) // 1996
>         at(year1996=0 year2012=1 agesq=2039) // 2012
4.         mlincom 1 - 2, add rowname(outcome=`iout')
5.         estimates restore go1m
6.     }
7. }
. mlincom
```

	lincom	pvalue	ll	ul
outcome=1	-0.009	0.000	-0.014	-0.004
outcome=2	-0.061	0.000	-0.094	-0.028
outcome=3	0.059	0.001	0.025	0.093
outcome=4	0.011	0.000	0.005	0.017

For details on the commands used, see section 7.12.1.

7.16.4 The sequential logit model

Some ordinal outcomes represent the progress of events or stages in some process through which an individual can advance. For example, the outcome could be faculty rank, where the stages are assistant professor, associate professor, and full professor. The key characteristic of the process is that an individual must pass through each stage. The outcome is thus the result of a sequence of potential transitions: an assistant professor may or may not make the transition to associate professor, and an associate professor may or may not make the transition to full professor.

The most straightforward way to model an outcome like this is as a series of BRMs. Consider the binary logit model from chapter 5:

$$\ln \frac{\Pr(y = 1 | \mathbf{x})}{\Pr(y = 0 | \mathbf{x})} = \alpha + \mathbf{x}\beta$$

where we have made the intercept explicit rather than including it in β . To extend this to multiple transitions, we estimate for each transition the log odds of having made the transition ($y > m$) versus not having made the transition ($y = m$). For example, we estimate the log odds of being an associate or a full professor ($y > 1$) versus being an assistant professor ($y = 1$). We allow separate coefficients β_m for each transition from $y = m$:

$$\ln \frac{\Pr(y > m | \mathbf{x})}{\Pr(y = m | \mathbf{x})} = \alpha_m + \mathbf{x}\beta_m \quad \text{for } m = 1 \text{ to } J - 1 \quad (7.5)$$

where J is the number of stages.

This is an example of a broader group of models called sequential logit models (for example, Liao [1994, 26–28]). This model differs importantly from the generalized ordered logit model in that observations in which $y < m$ are not used in the estimation of β_m . For example, assistant professors are not used when modeling the transition from associate professor to full professor.

To demonstrate how to fit this model, we use the variable `educ` in the `gssclass4` dataset as our outcome. The three values of `educ` represent two transitions: students may or may not graduate from high school, and high school graduates may or may not graduate from college. To fit the model, we first use `recode` to create dummy variables representing whether or not respondents at each stage made the transition to the next. The variable `educ` has the distribution

```
. tabulate educ, miss
```

educational attainment	Freq.	Percent	Cum.
not hs grad	993	17.67	17.67
hs only	3,270	58.19	75.85
college	1,357	24.15	100.00
Total	5,620	100.00	

We create the variable `gradcollege` to indicate if someone with a high school diploma graduated from college, where those who did not graduate from high school (`educ=1`) are recoded as missing, not as 0. Those who did not graduate from high school are not included in the analysis of the transition to college graduation.

```
. recode educ (1=0) (2 3=1), gen(gradhs)
(5620 differences between educ and gradhs)
. label var gradhs "Graduate high school?"
. recode educ (1=.) (2=0) (3=1), gen(gradcollege)
(5620 differences between educ and gradcollege)
. label var gradcollege "Graduate college?"
```

Next, we use `logit` to fit separate models for each transition, using race and sex as independent variables.

```
. * HS degree vs not
. logit gradhs i.white i.female, or nolog
Logistic regression
```

```
Number of obs   =      5620
LR chi2(2)      =      25.40
Prob > chi2     =      0.0000
Pseudo R2      =      0.0048
```

```
Log likelihood = -2608.1189
```

gradhs	Odds Ratio	Std. Err.	z	P> z	[95% Conf. Interval]	
white						
white	1.531009	.1280333	5.09	0.000	1.299554	1.803686
female						
female	.9665729	.0683325	-0.48	0.631	.8415082	1.110225
_cons	3.387367	.2880726	14.35	0.000	2.867301	4.001761

```
. * College degree vs HS degree
. logit gradcollege i.white i.female, or nolog
Logistic regression
```

```
Number of obs   =      4627
LR chi2(2)      =       8.81
Prob > chi2     =      0.0122
Pseudo R2      =      0.0016
```

```
Log likelihood = -2795.2139
```

gradcollege	Odds Ratio	Std. Err.	z	P> z	[95% Conf. Interval]	
white						
white	1.154414	.1007857	1.64	0.100	.9728539	1.369857
female						
female	.8564618	.0555153	-2.39	0.017	.7542818	.9724837
_cons	.4003614	.0352609	-10.39	0.000	.3368873	.4757949

We can interpret odds ratios for each of the transitions the same as we did in chapter 6. For example,

Being white compared with being nonwhite increases the odds of graduating high school by a factor of 1.53, holding gender constant.

The `seqlogit` command (Buis 2007) fits equations for all transitions simultaneously and provides the likelihood for the full model. The `seqlogit` command can also be used with `predict` and `margins`—and accordingly, with our `m*` commands—to compute probabilities of membership in each category. The syntax of `seqlogit` is complicated because it can be used to fit elaborate, branching sequences of transitions (see `help seqlogit` if installed). The `tree()` option is required and is used to specify how outcome values map onto different transitions. In our case, we specify `tree(1 : 2 3, 2 : 3)` because our two transitions are `educ==1` or `2` versus `educ==3` and `educ==2` versus `educ==3`:

```
. seqlogit educ i.white i.female, tree(1 : 2 3, 2 : 3) or nolog
Transition tree:
Transition 1: 1 : 2 3
Transition 2: 2 : 3
Computing starting values for:
Transition 1
Transition 2
```

					Number of obs	=	5620
					LR chi2(4)	=	34.21
					Prob > chi2	=	0.0000
Log likelihood = -5403.3329							

	educ	Odds Ratio	Std. Err.	z	P> z	[95% Conf. Interval]	
_2_3v1	white						
	white	1.531009	.1280333	5.09	0.000	1.299554	1.803686
	female						
	female	.9665729	.0683325	-0.48	0.631	.8415082	1.110225
	_cons	3.387367	.2880726	14.35	0.000	2.867301	4.001761
_3v2	white						
	white	1.154414	.1007857	1.64	0.100	.9728539	1.369857
	female						
	female	.8564618	.0555153	-2.39	0.017	.7542818	.9724837
	_cons	.4003614	.0352609	-10.39	0.000	.3368873	.4757949

```
. estimates store seqlogit
```


The coefficients here are precisely the same as when the equations were fit separately. The top equation presents coefficients for the transition to high school graduation (outcomes 2 and 3 versus 1) and the bottom for the transition to college graduate (outcome 3 versus 2). The log likelihood of the combined model fit with `seqlogit` is the sum of the log likelihoods of the separate binary logit models.

A more parsimonious model imposes the assumption that the coefficients for each independent variable are equal across all transitions. Instead of β_m in (7.5), we have the same β in all transition equations, while the intercepts differ:

$$\ln \frac{\Pr(y > m | \mathbf{x})}{\Pr(y = m | \mathbf{x})} = \alpha_m + \mathbf{x}\beta \quad \text{for } m = 1 \text{ to } J - 1$$

where J is the number of stages. This model is sometimes called the continuation ratio model and was first proposed by Fienberg (1980, 110).

The continuation ratio model can be fit using the user-written command `ocratio` (Wolfe 1998), although it uses a somewhat different parameterization than given here. Because `ocratio` is an older command, it does not support factor-variable notation and does not work with `predict`, `margins`, or our `m*` commands. The continuation ratio model can be fit with `seqlogit` if you impose the equality constraints on coefficients by using the `constraint define` command. This command defines constraints on a model's parameters that are imposed during estimation. Although further detail on constrained estimation is outside the scope of this book, we provide the example code below to show the use of the `constraint define` command and the `constraint()` option with `seqlogit`:

```
. constraint define 1 [_2_3v1]1.female=[_3v2]1.female
. constraint define 2 [_2_3v1]1.white=[_3v2]1.white
```

The key to understanding the constraints is understanding how `seqlogit` names the equations that it estimates. The equation comparing outcome 1 with outcomes 2 and 3 is named `_2_3v1` so that `[_2_3v1]1.female` indicates the coefficient for `1.female` in this equation. Accordingly, the constraint 1 defined above says that the coefficients for `1.female` are equal in both transition equations. Constraint 2 does the same thing for `1.white`. To impose these constraints during estimation, we add the option `constraint(1 2)` to the estimation command:


```
. seqlogit educ i.white i.female, tree(1: 2 3, 2 : 3) constraint(1 2) or nolog
Transition tree:
Transition 1: 1 : 2 3
Transition 2: 2 : 3
Computing starting values for:
Transition 1
Transition 2
```

Number of obs = 5620
Wald chi2(0) = .
Prob > chi2 = .

```
Log likelihood = -5406.6832
( 1) [_2_3v1]1.female - [_3v2]1.female = 0
( 2) [_2_3v1]1.white - [_3v2]1.white = 0
```

educ	Odds Ratio	Std. Err.	z	P> z	[95% Conf. Interval]	
_2_3v1						
white						
white	1.336865	.0824991	4.70	0.000	1.184566	1.508745
female						
female	.9046686	.0432715	-2.09	0.036	.8237122	.9935817
_cons	3.906983	.2600151	20.48	0.000	3.4292	4.451333
_3v2						
white						
white	1.336865	.0824991	4.70	0.000	1.184566	1.508745
female						
female	.9046686	.0432715	-2.09	0.036	.8237122	.9935817
_cons	.3437397	.0231741	-15.84	0.000	.3011923	.3922975

```
. estimates store contratio
```

The log likelihood is smaller than it was with the unconstrained, sequential logit model. Because the second model is nested within the first model, we can see whether the difference in fit is statistically significant with an LR test:

```
. lrtest contratio seqlogit
Likelihood-ratio test
(Assumption: contratio nested in seqlogit)
```

LR chi2(2) = 6.70
Prob > chi2 = 0.0351

The p -value is less than 0.05, so we reject the null hypothesis that the coefficients are equal across transitions. We could compare whether the coefficients for a particular independent variable are equal across transitions by constraining only those coefficients to be equal or by using `test` to compute a Wald test.

7.17 Conclusion

Ordinal outcomes are common, especially in survey research where respondents are presented with a question and a fixed set of categories with which to respond. The models we present are motivated by the premise of a latent, unidimensional continuum

that is divided by cutpoints into the observed categories because of the way the variable is measured. The ordered logit and ordered probit models thus follow nicely from their binary counterparts introduced in chapter 5, because we can think of a binary outcome the same way except with only one threshold. A key difference, however, is that with ordinal outcomes it is easy to imagine more than one dimension underlying responses or that independent variables differ in the magnitude of their influence over different thresholds. In our experience, users often come to ordinal models wondering when it is okay to treat ordinal categories as interval-level variables and use linear regression. One of the goals in this chapter is to push you to think also in the other direction, about whether an ostensibly ordinal outcome actually has properties that deserve more complex specifications, including perhaps even thinking of it as a nominal outcome. How to model nominal outcomes provides the focus of the next chapter.

8 Models for nominal outcomes

An outcome is nominal when the categories are assumed to be unordered. For example, marital status can be grouped nominally into the categories of divorced, never married, married, or widowed. Occupations might be organized as professional, white collar, blue collar, craft, and menial. Other examples include reasons for leaving the parents' home, the organizational context of scientific work such as industry, government, and academia, and the choice of language in a multilingual society. Further, in some cases you might prefer to treat an outcome as nominal even though it is ordered, ordered on multiple dimensions, or partially ordered. For example, if the response categories are strongly agree, agree, disagree, strongly disagree, and don't know, the category "don't know" invalidates models for ordinal outcomes. Or, you might decide to use a nominal regression model when the assumption of parallel regressions is rejected. In general, if you have concerns about the ordinality of the dependent variable, the potential loss of efficiency in using models for nominal outcomes is outweighed by avoiding potential bias.

In this chapter, we focus on the multinomial logit model (MNL), which is the most frequently used nominal regression model. This model essentially fits separate binary logits for each pair of outcome categories. Next, we consider the stereotype logistic regression model. Although this model is often used for ordinal outcomes, it is closely related to the MNL. These models assume that the data are case specific, meaning that each independent variable has one value for each individual. Examples of such variables are an individual's race or education. After that, we consider several models that include alternative-specific data. Alternative-specific variables vary not only by individual but also by the alternative. For example, if a commuter is selecting one of three modes of travel, an alternative-specific predictor might be her travel time using each alternative.

We use "alternative" to refer to a possible outcome. Sometimes we refer to an alternative as an outcome, a category, or a comparison group to be consistent with the usual terminology for a model or the output generated by Stata. The term "choice" refers to the alternative that is actually observed, which can be thought of as the "most preferred" alternative. For example, if the dependent variable is the party voted for in the last presidential election, the alternatives might be Republican, Democrat, and Independent. If a person voted for the alternative of Democrat, we would say that the choice for this case is Democrat. But you should not infer from the term "choice" that the models we describe can be used only for data where the outcome occurs through a process of choice. For example, if we were modeling the type of injuries that people

entering the emergency room of a hospital have, we would use the term “choice” even though the injury sustained is unlikely to be a choice.

We begin by discussing the MNLM, where the biggest challenge is that the model includes many parameters and so it is easy to be overwhelmed by the complexity of the results. This complexity is compounded by the nonlinearity of the model, which leads to the same difficulties of interpretation found for models in prior chapters. Although fitting the model is straightforward, interpretation involves challenges that are the focus of this chapter. We begin by reviewing the statistical model, followed by a discussion of testing, fit, and finally, methods of interpretation. For a more technical discussion of the model, see Long (1997). As discussed in chapter 1, you can obtain sample do-files and data files by downloading the `spost13.do` package.

The outcome for the primary example we have chosen for this chapter is political party affiliation, collected from a survey that used the categories strong Democrat; Democrat, Independent, Republican; and strong Republican. Although this variable may initially appear to be ordinal, our analysis suggests that it is ordered on two dimensions relative to the independent variables we consider. On the attribute of left-right orientation, the categories increase from strong Democrat to strong Republican. In terms of intensity of partisanship, the categories are ordered Independent to either Republican or Democrat and then either strong Republican or strong Democrat. This violates Stevens' (1946) definition of an ordinal scale as a variable that uses numbers to indicate rank ordering on a single attribute. Indeed, when you use an ordinal model, we recommend also fitting the model using multinomial logit as a sensitivity analysis.

8.1 The multinomial logit model

The MNLM can be thought of as simultaneously fitting binary logits for all comparisons among the alternatives. For example, let `party3` be a categorical variable with the outcomes *D* for Democrat, *I* for Independent, and *R* for Republican.¹ Assume that there is one independent variable measuring income in \$1,000s. We can examine the effect of `income` on `party3` by fitting three binary logits,

$$\begin{aligned}\ln \frac{\Pr(D | \mathbf{x})}{\Pr(I | \mathbf{x})} &= \beta_{0,D|I} + \beta_{1,D|I} \text{income} \\ \ln \frac{\Pr(R | \mathbf{x})}{\Pr(I | \mathbf{x})} &= \beta_{0,R|I} + \beta_{1,R|I} \text{income} \\ \ln \frac{\Pr(D | \mathbf{x})}{\Pr(R | \mathbf{x})} &= \beta_{0,D|R} + \beta_{1,D|R} \text{income}\end{aligned}$$

where the subscripts to the β 's indicate which comparison is being made. For example, $\beta_{1,D|I}$ is the coefficient for the first independent variable for the comparison of *D* and *I*.

1. Variable `party3` combines categories `StrDem` and `Dem` and the categories `Rep` and `StrRep` from the variable `party` used later in the chapter.

The three binary logits include redundant information. Because $\ln a/b = \ln a - \ln b$, the following equality must hold:

$$\ln \frac{\Pr(D | \mathbf{x})}{\Pr(I | \mathbf{x})} - \ln \frac{\Pr(R | \mathbf{x})}{\Pr(I | \mathbf{x})} = \ln \frac{\Pr(D | \mathbf{x})}{\Pr(R | \mathbf{x})}$$

This implies that

$$\beta_{0,D|I} - \beta_{0,R|I} = \beta_{0,D|R} \quad (8.1)$$

$$\beta_{1,D|I} - \beta_{1,R|I} = \beta_{1,D|R}$$

In general, with J alternatives, only $J - 1$ binary logits need to be fit. These $J - 1$ logits are referred to as a minimal set. Estimates for the remaining coefficients can be computed using equalities of the sort shown in (8.1).

Fitting the MNLM by fitting a series of binary logits is not optimal because each binary logit is based on a different sample. For example, in the logit comparing D with I , those in R are dropped. To see this, we begin by loading the data and examining the distribution of party3:

```
. use partyid4, clear
(partyid4.dta | 1992 American National Election Study | 2014-03-12)
. tabulate party3
```

Party ID	Freq.	Percent	Cum.
Democrat	693	50.14	50.14
Independent	151	10.93	61.07
Republican	538	38.93	100.00
Total	1,382	100.00	

Next, we fit a binary logit model comparing Democrats with Independents by using the dependent variable dem_ind:

```
. tabulate dem_ind, miss
```

Democrat or Independent	Freq.	Percent	Cum.
Independent	151	10.93	10.93
Democrat	693	50.14	61.07
.	538	38.93	100.00
Total	1,382	100.00	

```
. logit dem_ind income, nolog
```

Logistic regression

```
Number of obs =      844
LR chi2(1)     =       0.48
Prob > chi2    =      0.4873
Pseudo R2     =      0.0006
```

Log likelihood = -396.21646

dem_ind	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
income	-.0024887	.0035513	-0.70	0.483	-.009449 .0044717
_cons	1.605464	.1485698	10.81	0.000	1.314273 1.896656

The logit includes only 844 cases out of the sample of 1,382 because those who are Republican are excluded as missing. Next, we fit a binary logit comparing Republicans and Independents, excluding Democrats from the sample:

```
. tabulate rep_ind, miss
```

Republican or Independent	Freq.	Percent	Cum.
Independent	151	10.93	10.93
Republican	538	38.93	49.86
.	693	50.14	100.00
Total	1,382	100.00	

```
. logit rep_ind income, nolog
```

Logistic regression

Number of obs = 689
LR chi2(1) = 20.41
Prob > chi2 = 0.0000
Pseudo R2 = 0.0282

Log likelihood = -352.09947

rep_ind	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
income	.0156761	.0037443	4.19	0.000	.0083374 .0230148
_cons	.6585897	.1624946	4.05	0.000	.3401061 .9770732

And last, we exclude Independents:

```
. tabulate dem_rep, miss
```

Democrat or Republican	Freq.	Percent	Cum.
Republican	538	38.93	38.93
Democrat	693	50.14	89.07
.	151	10.93	100.00
Total	1,382	100.00	

```
. logit dem_rep income, nolog
```

Logistic regression

Number of obs = 1231
LR chi2(1) = 71.89
Prob > chi2 = 0.0000
Pseudo R2 = 0.0426

Log likelihood = -807.53617

dem_rep	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
income	-.0183709	.0022958	-8.00	0.000	-.0228706 -.0138712
_cons	.9525295	.1046759	9.10	0.000	.7473684 1.157691

The results from the binary logits can be compared with those obtained by fitting the MNLM with `mlogit`:

```
. mlogit party3 income, nolog base(2)
Multinomial logistic regression
Log likelihood = -1283.3075
Number of obs   =    1382
LR chi2(2)      =    73.84
Prob > chi2     =    0.0000
Pseudo R2      =    0.0280
```

party3	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
Democrat						
income	-.002724	.0037162	-0.73	0.464	-.0100077	.0045597
_cons	1.613193	.1529897	10.54	0.000	1.313338	1.913047
Independent	(base outcome)					
Republican						
income	.0151864	.0036605	4.15	0.000	.0080119	.022361
_cons	.6779478	.1597457	4.24	0.000	.3648519	.9910437

The output is divided into three panels. The top panel is labeled **Democrat**, which is the value label for the first outcome category of the dependent variable; the second panel is labeled **Independent**, which is the base outcome that we discuss shortly; and the third panel corresponds to the third outcome, **Republican**. The coefficients in the first and third panels are for comparisons with the base outcome, **Independent**. Thus the panel **Democrat** shows estimates of coefficients from the comparison of *D* with the base *I*, while the panel **Republican** holds the estimates comparing *R* with *I*. Accordingly, the top panel should be compared with the coefficients from the binary logit for *D* and *I* (outcome variable `dem_ind`). For example, the estimate for the comparison of *D* with *I* from `mlogit` is $\hat{\beta}_{1,D|I} = -0.002724$ with $z = -0.73$, whereas the `logit` estimate is $\hat{\beta}_{1,D|I} = -0.0024887$ with $z = -0.70$. Overall, the estimates from the binary model are close to those from the MNLM but not exactly the same.

Although theoretically $\beta_{1,D|I} - \beta_{1,R|I} = \beta_{1,D|R}$, the estimates from the binary logits are $\hat{\beta}_{1,D|I} - \hat{\beta}_{1,R|I} = (-0.0024887) - (0.0156761) = -0.0181648$, which do not quite equal the binary logit estimate $\hat{\beta}_{1,D|R} = -0.0183709$. This occurs because a series of binary logits fit with `logit` does not impose the constraints among coefficients that are implicit in the definition of the MNLM. When fitting the model with `mlogit`, these constraints are imposed. Indeed, the output from `mlogit` presents only two of the three comparisons from our example, namely, *D* versus *I* and *R* versus *I*. The remaining comparison, *D* versus *R*, is exactly equal to the difference between the two sets of estimated coefficients. The critical point here is simple:

The MNLM may be understood as a set of binary logits among all pairs of outcomes.

8.1.1 Formal statement of the model

The MNLM can be written as

$$\ln \Omega_{m|b}(\mathbf{x}) = \ln \frac{\Pr(y = m | \mathbf{x})}{\Pr(y = b | \mathbf{x})} = \mathbf{x}\beta_{m|b} \quad \text{for } m = 1 \text{ to } J$$

where b is the base outcome, sometimes called the reference category. As $\ln \Omega_{b|b}(\mathbf{x}) = \ln 1 = 0$, it follows that $\beta_{b|b} = 0$. That is, the log odds of an outcome compared with itself is always 0, and thus the effects of any independent variables must also be 0.

These J equations can be solved to compute the probabilities for each outcome:

$$\Pr(y = m | \mathbf{x}) = \frac{\exp(\mathbf{x}\beta_{m|b})}{\sum_{j=1}^J \exp(\mathbf{x}\beta_{j|b})}$$

The probabilities will be the same regardless of the base outcome b that is used. For example, suppose that you have three outcomes and fit the model with alternative 1 as the base, where you would obtain estimates $\hat{\beta}_{2|1}$ and $\hat{\beta}_{3|1}$, with $\beta_{1|1} = 0$. The probability equation is

$$\Pr(y = m | \mathbf{x}) = \frac{\exp(\mathbf{x}\beta_{m|1})}{\sum_{j=1}^J \exp(\mathbf{x}\beta_{j|1})}$$

If someone else set up the model with base outcome 2, they would obtain estimates $\hat{\beta}_{1|2}$ and $\hat{\beta}_{3|2}$, with $\beta_{2|2} = 0$. Their probability equation would be

$$\Pr(y = m | \mathbf{x}) = \frac{\exp(\mathbf{x}\beta_{m|2})}{\sum_{j=1}^J \exp(\mathbf{x}\beta_{j|2})}$$

The estimated parameters are different because they are estimating different things, but they produce exactly the same predictions. Confusion arises only if you are not clear about which parameterization you are using. We return to this issue when we discuss how Stata's `mlogit` parameterizes the model in the next section.

8.2 Estimation using the `mlogit` command

The MNLM is fit with the following command and its basic options:

```
mlogit depvar [indepvars] [if] [in] [weight] [, noconstant
    baseoutcome(#) vce(vcetype) rrr ]
```

For other options, run `help mlogit`. In our experience, the model converges quickly, even when there are many outcome categories and independent variables.

Variable lists

depvar is the dependent variable. The specific values taken on by the dependent variable are irrelevant as long as they are integers. For example, if you had three outcomes, you could use the values 1, 2, and 3 or -1, 0, and 999. Nevertheless, to avoid confusion, we strongly recommend coding your outcome as consecutive integers beginning with 1.

indepvars is a list of independent variables. If *indepvars* is not included, Stata fits a model with only constants.

Specifying the estimation sample

if and in qualifiers. *if* and *in* qualifiers can be used to restrict the estimation sample. For example, if you want to fit the model with only white respondents, use the command `mlogit party i.educ income10 if black==0`.

Listwise deletion. Stata excludes cases in which there are missing values for any of the variables. Accordingly, if two models are fit using the same dataset but have different sets of independent variables, it is possible to have different samples. We recommend that you use `mark` and `markout` (discussed in chapter 3) to explicitly remove cases with missing data.

Weights and complex samples

`mlogit` can be used with *fweights*, *pweights*, and *iwweights*. Survey estimation is supported. See chapter 3 for details.

Options

`noconstant` excludes the constant terms from the model.

`baseoutcome(#)` specifies the value of *depvar* that is the base outcome (that is, reference group) for the coefficients that are listed. This determines how the model is parameterized. If `baseoutcome()` is not specified, the most frequent outcome in the estimation sample is used as the base. The base is reported as (base outcome) in the table of estimates.

`vce(vcetype)` specifies the type of standard errors to be computed. See section 3.1.9 for details.

`rrr` reports the estimated coefficients transformed to relative-risk ratios, defined as $\exp(b)$ rather than b , along with standard errors and confidence intervals for these ratios. Relative risk ratios are also referred to as odds ratios.

8.2.1 Example of MNLM

The 1992 American National Election Study asked respondents to indicate their political party using one of eight categories. We used these to create `party` with five categories: strong Democrat (1 = `StrDem`), Democrat (2 = `Dem`), Independent (3 = `Indep`), Republican (4 = `Rep`), and strong Republican (5 = `StrRep`):

```
. tabulate party, miss
```

Party ID	Freq.	Percent	Cum.
StrDem	266	19.25	19.25
Dem	427	30.90	50.14
Indep	151	10.93	61.07
Rep	369	26.70	87.77
StrRep	169	12.23	100.00
Total	1,382	100.00	

To simplify our notation, at times we abbreviate `StrDem` as `SD`, `Dem` as `D`, `Indep` as `I`, `Rep` as `R`, and `StrRep` as `SR`.

Five regressors are included in the model: age, income, race (indicated as black or not), gender, and education (measured as not completing high school, completing high school but not college, and completing college). Descriptive statistics for the continuous and binary variables are

```
. sum age income black female
```

Variable	Obs	Mean	Std. Dev.	Min	Max
age	1382	45.94645	16.78311	18	91
income	1382	37.45767	27.78148	1.5	131.25
black	1382	.1374819	.34448	0	1
female	1382	.4934877	.5001386	0	1

The distribution of educational attainment is

```
. tabulate educ, miss
```

Level of education	Freq.	Percent	Cum.
not hs grad	222	16.06	16.06
hs only	802	58.03	74.10
college	358	25.90	100.00
Total	1,382	100.00	

Using these variables, we fit the model

```
. mlogit party age income i.black i.female i.educ
(output omitted)
```


Because `educ` has three categories, `i.educ` is expanded to `2.educ` and `3.educ`, leading to the following minimal set of equations:

$$\begin{aligned}\ln \Omega_{SD|SR}(\mathbf{x}_i) &= \beta_{0,SD|SR} + \beta_{1,SD|SR}\text{age} + \beta_{2,SD|SR}\text{income} + \beta_{3,SD|SR}\text{black} \\ &\quad + \beta_{4,SD|SR}\text{female} + \beta_{5,SD|SR}2.\text{educ} + \beta_{6,SD|SR}3.\text{educ} \\ \ln \Omega_{D|SR}(\mathbf{x}_i) &= \beta_{0,D|SR} + \beta_{1,D|SR}\text{age} + \beta_{2,D|SR}\text{income} + \beta_{3,D|SR}\text{black} \\ &\quad + \beta_{4,D|SR}\text{female} + \beta_{5,D|SR}2.\text{educ} + \beta_{6,D|SR}3.\text{educ} \\ \ln \Omega_{I|SR}(\mathbf{x}_i) &= \beta_{0,I|SR} + \beta_{1,I|SR}\text{age} + \beta_{2,I|SR}\text{income} + \beta_{3,I|SR}\text{black} \\ &\quad + \beta_{4,I|SR}\text{female} + \beta_{5,I|SR}2.\text{educ} + \beta_{6,I|SR}3.\text{educ} \\ \ln \Omega_{R|SR}(\mathbf{x}_i) &= \beta_{0,R|SR} + \beta_{1,R|SR}\text{age} + \beta_{2,R|SR}\text{income} + \beta_{3,R|SR}\text{black} \\ &\quad + \beta_{4,R|SR}\text{female} + \beta_{5,R|SR}2.\text{educ} + \beta_{6,R|SR}3.\text{educ}\end{aligned}$$

where the fifth outcome, `StrRep`, is the base. The (lengthy) results are

```
. mlogit party age income i.black i.female i.educ, base(5) vsquish
Iteration 0:  log likelihood = -2116.5357
Iteration 1:  log likelihood = -1973.8136
Iteration 2:  log likelihood = -1961.2327
Iteration 3:  log likelihood = -1960.9125
Iteration 4:  log likelihood = -1960.9107
Iteration 5:  log likelihood = -1960.9107
```


Multinomial logistic regression

Number of obs = 1382
 LR chi2(24) = 311.25
 Prob > chi2 = 0.0000
 Pseudo R2 = 0.0735

Log likelihood = -1960.9107

party	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
StrDem						
age	.0028185	.00644	0.44	0.662	-.0098036	.0154407
income	-.0174695	.0045777	-3.82	0.000	-.0264416	-.0084974
black						
yes	3.075438	.604052	5.09	0.000	1.891518	4.259358
female						
yes	.2368373	.215026	1.10	0.271	-.1846059	.6582805
educ						
hs only	-.5548853	.3426066	-1.62	0.105	-1.226382	.1166113
college	-1.374585	.3990504	-3.44	0.001	-2.156709	-.5924606
_cons	1.182225	.5132429	2.30	0.021	.1762875	2.188163
Dem						
age	-.0207981	.0059291	-3.51	0.000	-.032419	-.0091772
income	-.0101908	.0035532	-2.87	0.004	-.0171549	-.0032267
black						
yes	2.07911	.6030684	3.45	0.001	.8971176	3.261102
female						
yes	.4776808	.1915945	2.49	0.013	.1021624	.8531992
educ						
hs only	-.2097834	.3365993	-0.62	0.533	-.8695059	.4499392
college	-.7459487	.3691435	-2.02	0.043	-1.469457	-.0224408
_cons	2.332098	.4744766	4.92	0.000	1.402141	3.262055
Indep						
age	-.0287992	.0074315	-3.88	0.000	-.0433648	-.0142337
income	-.0089716	.0047821	-1.88	0.061	-.0183443	.0004012
black						
yes	2.290928	.6262902	3.66	0.000	1.063422	3.518435
female						
yes	.0478994	.2361813	0.20	0.839	-.4150074	.5108062
educ						
hs only	-.6018342	.3788561	-1.59	0.112	-1.344379	.1407101
college	-1.758295	.4510057	-3.90	0.000	-2.64225	-.8743398
_cons	2.225948	.553075	4.02	0.000	1.141941	3.309955
Rep						
age	-.0217144	.0060422	-3.59	0.000	-.0335569	-.0098718
income	-.0012715	.0033629	-0.38	0.705	-.0078627	.0053196
black						
yes	.106285	.6861838	0.15	0.877	-1.238611	1.451181
female						
yes	.244697	.1929118	1.27	0.205	-.1334031	.6227971
educ						
hs only	-.1827121	.3502744	-0.52	0.602	-.8692374	.5038132
college	-.6956311	.3804257	-1.83	0.067	-1.441252	.0499896
_cons	2.092855	.4846516	4.32	0.000	1.142955	3.042754
StrRep	(base outcome)					

Methods of testing coefficients and interpretation of the estimates will be considered after we discuss the effects of selecting different base outcomes.

8.2.2 Selecting different base outcomes

By default, `mlogit` sets the base outcome to the alternative with the most observations in the estimation sample. Or, as illustrated in the last example, you can select the base with the option `baseoutcome()`, which can be abbreviated simply as `b()`. `mlogit` reports the coefficients for each independent variable on each outcome relative to the base outcome.

Although `mlogit` only shows coefficients comparing outcomes with the base outcome, it is important to examine the coefficients for other pairs of outcomes. For example, you might be interested in the effect of being a female on being `Dem` compared with `Indep` (that is, $\beta_{\text{female}, D|I}$), which was not estimated in the output above. Although this coefficient can be estimated by running `mlogit` with a different base outcome (for example, `mlogit party ..., base(3)`), it is easier to use `listcoef`, which presents estimates for all pairs of outcome categories. Because `listcoef` can generate lengthy output, we illustrate several options that limit which coefficients are listed. First, if you specify a list of variables, only coefficients for those variables are shown. For example,

```
. listcoef female, help
```

```
mlogit (N=1382): Factor change in the odds of party
```

```
Variable: 1.female (sd=0.500)
```

		b	z	P> z	e ^b	e ^b StdX
StrDem	vs Dem	-0.2408	-1.443	0.149	0.786	0.887
StrDem	vs Indep	0.1889	0.889	0.374	1.208	1.099
StrDem	vs Rep	-0.0079	-0.044	0.965	0.992	0.996
StrDem	vs StrRep	0.2368	1.101	0.271	1.267	1.126
Dem	vs StrDem	0.2408	1.443	0.149	1.272	1.128
Dem	vs Indep	0.4298	2.217	0.027	1.537	1.240
Dem	vs Rep	0.2330	1.587	0.112	1.262	1.124
Dem	vs StrRep	0.4777	2.493	0.013	1.612	1.270
Indep	vs StrDem	-0.1889	-0.889	0.374	0.828	0.910
Indep	vs Dem	-0.4298	-2.217	0.027	0.651	0.807
Indep	vs Rep	-0.1968	-0.983	0.326	0.821	0.906
Indep	vs StrRep	0.0479	0.203	0.839	1.049	1.024
Rep	vs StrDem	0.0079	0.044	0.965	1.008	1.004
Rep	vs Dem	-0.2330	-1.587	0.112	0.792	0.890
Rep	vs Indep	0.1968	0.983	0.326	1.217	1.103
Rep	vs StrRep	0.2447	1.268	0.205	1.277	1.130
StrRep	vs StrDem	-0.2368	-1.101	0.271	0.789	0.888
StrRep	vs Dem	-0.4777	-2.493	0.013	0.620	0.787
StrRep	vs Indep	-0.0479	-0.203	0.839	0.953	0.976
StrRep	vs Rep	-0.2447	-1.268	0.205	0.783	0.885

b = raw coefficient

z = z-score for test of b=0

P>|z| = p-value for z-test

e^b = exp(b) = factor change in odds for unit increase in X

e^bStdX = exp(b*SD of X) = change in odds for SD increase in X

Notice that none of the coefficients with the base outcome **StrDem** are statistically significant, even at the 0.10 level. Although these are the only coefficients that are shown in the output from **mlogit**, two of the coefficients relative to outcome **Dem** are significant at the 0.05 level and two more are almost significant at the 0.10 level. Before concluding that a variable has no effect, you should examine all the contrasts and compute an omnibus test for the effect of a variable, as discussed in section 8.3.2.

By default, **listcoef** shows coefficients for all contrasts. For example, it even shows you the coefficient comparing **StrRep** with **Rep**, which equals -0.2447 , and the coefficient comparing **Rep** with **StrRep**, which equals 0.2447 . You can limit which contrasts are shown with the **gt**, **lt**, or **adjacent** options. With the **gt** option, only coefficients in which the category number of the first alternative is greater than that of the second are shown; **lt** shows comparisons when the first alternative is less than the second; and **adjacent** limits coefficients to those from adjacent outcomes. For example,

```
. listcoef income age, lt adjacent
```

```
mlogit (N=1382): Factor change in the odds of party
```

```
Variable: age (sd=16.783)
```

		b	z	P> z	e ^{-b}	e ^{-b} StdX
StrDem	vs Dem	0.0236	4.761	0.000	1.024	1.486
Dem	vs Indep	0.0080	1.287	0.198	1.008	1.144
Indep	vs Rep	-0.0071	-1.099	0.272	0.993	0.888
Rep	vs StrRep	-0.0217	-3.594	0.000	0.979	0.695

```
Variable: income (sd=27.781)
```

		b	z	P> z	e ^{-b}	e ^{-b} StdX
StrDem	vs Dem	-0.0073	-1.777	0.075	0.993	0.817
Dem	vs Indep	-0.0012	-0.279	0.780	0.999	0.967
Indep	vs Rep	-0.0077	-1.778	0.075	0.992	0.807
Rep	vs StrRep	-0.0013	-0.378	0.705	0.999	0.965

We will return to the contrasting patterns of coefficients for these two variables when we present methods for plotting coefficients in section 8.11.2.

You can also restrict the list of contrasts shown to only those with positive coefficients by using the **positive** option. This means you will see all the contrasts that are not simply derived by reversing the sign of another contrasts. This can often allow you to see at a glance overall patterns in the direction of the relationships between an independent variable and the outcome, as in this example:


```
. listcoef income, positive
mlogit (N=1382): Factor change in the odds of party
Variable: income (sd=27.781)
```

		b	z	P> z	e ^b	e ^b StdX
Dem	vs StrDem	0.0073	1.777	0.075	1.007	1.224
Indep	vs StrDem	0.0085	1.652	0.098	1.009	1.266
Indep	vs Dem	0.0012	0.279	0.780	1.001	1.034
Rep	vs StrDem	0.0162	3.916	0.000	1.016	1.568
Rep	vs Dem	0.0089	3.026	0.002	1.009	1.281
Rep	vs Indep	0.0077	1.778	0.075	1.008	1.239
StrRep	vs StrDem	0.0175	3.816	0.000	1.018	1.625
StrRep	vs Dem	0.0102	2.868	0.004	1.010	1.327
StrRep	vs Indep	0.0090	1.876	0.061	1.009	1.283
StrRep	vs Rep	0.0013	0.378	0.705	1.001	1.036

From the example, we can see that the positive coefficients for the income variable all correspond to contrasts of a further right outcome to a further left one. In other words, we can see that income is positive related to selecting a partisan identity further to the right. This corresponds to the conclusion that partisan identification behaves like an ordinal variable with respect to income (but, as we will show later, this is not the case for all the variables in our model.)

Finally, a last way to restrict the list of coefficients is with the `pvalue(#)` option that restricts coefficients to those that are significant at the specified level. For example,

```
. listcoef female, pvalue(.15)
mlogit (N=1382): Factor change in the odds of party (P<0.15)
Variable: 1.female (sd=0.500)
```

		b	z	P> z	e ^b	e ^b StdX
StrDem	vs Dem	-0.2408	-1.443	0.149	0.786	0.887
Dem	vs StrDem	0.2408	1.443	0.149	1.272	1.128
Dem	vs Indep	0.4298	2.217	0.027	1.537	1.240
Dem	vs Rep	0.2330	1.587	0.112	1.262	1.124
Dem	vs StrRep	0.4777	2.493	0.013	1.612	1.270
Indep	vs Dem	-0.4298	-2.217	0.027	0.651	0.807
Rep	vs Dem	-0.2330	-1.587	0.112	0.792	0.890
StrRep	vs Dem	-0.4777	-2.493	0.013	0.620	0.787

Using these options can reduce the amount of output from `listcoef` and focus attention on the most important results.

8.2.3 Predicting perfectly

The `mlogit` command handles perfect prediction in the same way as the `ologit` and `oprobit` commands, but somewhat differently than estimation commands for binary models. `logit` and `probit` automatically remove the observations that imply perfect prediction and compute the estimates accordingly. `mlogit` and `oprobit` keep these ob-

servations in the model, set the z statistics for the problem variables to 0, warn that standard errors are questionable, and indicate that a given number of observations are completely determined. You should refit the model after excluding the problem variable and deleting the observations that imply the perfect predictions. Using `tabulate` to cross-tabulate the problem variable and the dependent variable should reveal the combination of values that results in perfect prediction.²

8.3 Hypothesis testing

In the MNLM, you can test individual coefficients with the reported z statistics, with a Wald test by using `test`, or with an LR test by using `lrtest`. As the methods of testing a single coefficient that were discussed in chapters 3, 5, and 7 apply fully, they are not considered further here. However, in the MNLM, there are new reasons for testing sets of coefficients. First, testing that a variable has no effect requires a test that $J - 1$ coefficients in a minimal set are simultaneously equal to 0. Second, testing whether the independent variables as a group differentiate between two alternatives requires a test of K coefficients, where K is the number of independent variables, including those created by expanding factor-variable notation. In this section, we focus on these two kinds of tests.

Caution regarding specification searches. Given the difficulties of interpretation that are associated with the MNLM, it is tempting to search for a more parsimonious model by excluding variables or combining outcome categories based on a sequence of tests. Such a search requires great care. First, these tests involve multiple coefficients. Although the overall test might indicate that as a group the coefficients are not significantly different from 0, an individual coefficient could still be substantively and statistically significant. Accordingly, you should examine the individual coefficients involved in each test before deciding to revise your model. Second, as with all searches that use repeated, sequential tests, there is a danger of overfitting the model to the data. Whenever model specifications are determined based on prior testing using the same data, significance levels should be used only as rough guidelines.

8.3.1 `mlogtest` for tests of the MNLM

Although the tests in this section can be computed using `test` or `lrtest`, in practice this is tedious. The `mlogtest` command by Freese and Long (2000) makes the computation of these tests easy. The syntax is

2. Before Stata 13.1, `mlogit` produced the same output but did not provide a warning.


```
mlogtest [varlist] [, lr wald set([setname=]varlist
    [\ [setname=]varlist ][\ ...]) combine lrcombine iia hausman smhsiao
    detail base all]
```

varlist indicates the variables for which tests of significance should be computed. If no *varlist* is given, tests are run for all independent variables.

Options

lr requests a likelihood-ratio (LR) test for each variable in *varlist*. If *varlist* is not specified, tests for all variables are computed.

wald requests a Wald test for each variable in *varlist*. If *varlist* is not specified, tests for all variables are computed.

set([*setname1*=]*varlist1* [\ [*setname2*=]*varlist2*][\ ...]) specifies that a set of variables be considered together for the LR test or Wald test. \ is used to indicate that a new set of variables is being specified. For example, **mlogtest, lr set(age income \ 2.educ 3.educ)** computes one LR test for the hypothesis that the effects of *age* and *income* are jointly 0 and a second LR test that the effects of *2.educ* and *3.educ* are jointly 0. The option **set()** is used to label the output.

combine requests Wald tests of whether dependent categories can be combined.

lrcombine requests LR tests of whether dependent categories can be combined. These tests use constrained estimation and overwrite constraint 999 if it is already defined.

For other options, type **help mlogtest**.

8.3.2 Testing the effects of the independent variables

With J dependent categories, there are $J - 1$ nonredundant coefficients associated with each independent variable x_k . For example, in our model of party affiliation, there are four coefficients associated with *female*: $\beta_{\text{female},SD|SR}$, $\beta_{\text{female},D|SR}$, $\beta_{\text{female},I|SR}$, and $\beta_{\text{female},R|SR}$. The hypothesis that x_k does not affect the dependent variable can be written as

$$H_0: \beta_{k,1|b} = \cdots = \beta_{k,J|b} = 0$$

where b is the base outcome. Because $\beta_{k,b|b}$ is necessarily 0, the hypothesis imposes constraints on $J - 1$ parameters. This hypothesis can be tested with either a Wald or an LR test.

Likelihood-ratio test

The LR test involves 1) fitting the full model that includes all the variables, resulting in the LR statistic $\text{LR } \chi_F^2$; 2) fitting the restricted model that excludes variable x_k , resulting in $\text{LR } \chi_R^2$; and 3) computing the difference $\text{LR } \chi_{RvsF}^2 = \text{LR } \chi_F^2 - \text{LR } \chi_R^2$, which

is distributed as chi-squared with $J-1$ degrees of freedom if the null hypothesis is true. This can be done using `lrtest` by first fitting the full model and storing the estimates:

```
. mlogit party age income i.black i.female i.educ, base(5) nolog
(output omitted)
. estimates store full_model
```

Next, we fit a model that drops the variable `age`, again storing the estimates:

```
. mlogit party income i.black i.female i.educ, base(5) nolog
. estimates store drop_age
```

Finally, we compute the LR test:

```
. lrtest full_model drop_age
Likelihood-ratio test                                LR chi2(4) =    45.16
(Assumption: drop_age nested in full_model)          Prob > chi2 =    0.0000
```

Although using `lrtest` is straightforward, the command `mlogtest`, `lr` is even simpler because it automatically fits the needed models and computes the tests for all variables by making repeated calls to `lrtest`:

```
. mlogit party age income i.black i.female i.educ, base(5) nolog
(output omitted)
. mlogtest, lr
```

LR tests for independent variables (N=1382)

Ho: All coefficients associated with given variable(s) are 0

	chi2	df	P>chi2
age	45.165	4	0.000
income	24.361	4	0.000
1.black	126.467	4	0.000
1.female	9.143	4	0.058
2.educ	5.567	4	0.234
3.educ	21.582	4	0.000

The results of the LR test, regardless of how they are computed, can be interpreted as follows:

The effect of age on party affiliation is significant at the 0.01 level ($LR \chi^2 = 45.17$, $df = 4$, $p < 0.01$).

The effect of being female is significant at the 0.10 level but not at the 0.05 level ($LR \chi^2 = 9.14$, $df = 4$, $p = 0.06$).

This can also be stated more formally:

The hypothesis that all the coefficients associated with income are simultaneously equal to 0 can be rejected at the 0.01 level ($LR \chi^2 = 24.36$, $df = 4$, $p < 0.01$).

Wald test

Although we consider the LR test superior, its computational costs can be prohibitive if the model is complex or the sample is very large. Also, LR tests cannot be used if robust standard errors or survey estimation is used. Wald tests are computed using `test` and can be used with robust standard errors and survey estimation. As an example, to compute a Wald test of the null hypothesis that the effect of being female is 0, type

```
. mlogit party age income i.black i.female i.educ, base(5) nolog
(output omitted)
. test 1.female
( 1) [StrDem]1.female = 0
( 2) [Dem]1.female = 0
( 3) [Indep]1.female = 0
( 4) [Rep]1.female = 0
( 5) [StrRep]10.female = 0
Constraint 5 dropped
      chi2( 4) =    9.09
Prob > chi2 =   0.0590
```

The output from `test` makes explicit which coefficients are being tested and shows how Stata labels parameters in models with multiple equations. For example, the first line, labeled `[StrDem]1.female`, refers to the coefficient for `female` in the equation comparing the outcome `StrDem` with the base outcome `StrRep`; `[Dem]1.female` is the coefficient for `female` in the equation comparing the outcome `Dem` with the base category `StrRep`; and so on. The fifth constraint, listed as `[StrRep]10.female = 0`, refers to the coefficient comparing `StrRep` to `StrRep`, which is automatically constrained to 0 when the model is fit. The `10` in `10.female` means that the coefficient for outcome 1 was omitted in the model and so the parameter was not estimated. Accordingly, this constraint is dropped. The following command automates this process:

```
. mlogtest, wald
Wald tests for independent variables (N=1382)
Ho: All coefficients associated with given variable(s) are 0
```

	chi2	df	P>chi2
age	43.815	4	0.000
income	22.985	4	0.000
i.black	83.978	4	0.000
1.female	9.087	4	0.059
2.educ	5.569	4	0.234
3.educ	20.613	4	0.000

These tests can be interpreted in the same way as shown for the LR test above.

Testing multiple independent variables

The logic of the Wald or LR tests can be extended to test that the effects of two or more independent variables are simultaneously 0. For example, the hypothesis to test that x_k and x_ℓ have no effects is

$$H_0: \beta_{k,1|b} = \dots = \beta_{k,J|b} = \beta_{\ell,1|b} = \dots = \beta_{\ell,J|b} = 0$$

For example, to test the hypothesis that the effects of `age` and `income` are simultaneously equal to 0, we could use `lrtest` as follows:

```
. mlogit party age income i.black i.female i.educ, base(5) nolog
(output omitted)
. estimates store full_model
. mlogit party i.black i.female i.educ, base(5) nolog
(output omitted)
. estimates store drop_ageinc
. lrtest full_model drop_ageinc
Likelihood-ratio test                                LR chi2(8) =      71.58
(Assumption: drop_ageinc nested in full_model)        Prob > chi2 =    0.0000
```

We can use the `set` option in `mlogtest` to do the same things. Suppose we use the command

```
mlogtest, lr set(age&inc=age income \ educ=2.educ 3.educ)
```

The argument `age&inc=age income` specifies a test that the coefficients for `age` and `income` are simultaneously 0, labeling the results with the tag `age&inc`. Following the `\`, we specify a test that the coefficients for all the indicators of the factor variable `educ` are 0. Here are the results:

```
. mlogit party age income i.black i.female i.educ, base(5) nolog
(output omitted)
. mlogtest, lr set(age&inc=age income \ educ=2.educ 3.educ)
LR tests for independent variables (N=1382)
Ho: All coefficients associated with given variable(s) are 0
```

	chi2	df	P>chi2
age	45.165	4	0.000
income	24.361	4	0.000
1.black	126.467	4	0.000
1.female	9.143	4	0.058
2.educ	5.567	4	0.234
3.educ	21.582	4	0.000
age&inc	71.585	8	0.000
educ	26.881	8	0.001

```
age&inc contains: age income
educ contains: 2.educ 3.educ
```


8.3.3 Tests for combining alternatives

If none of the independent variables significantly affects the odds of alternative m versus alternative n , we may say that m and n are indistinguishable with respect to the variables in the model (Anderson 1984). To say that alternatives m and n are indistinguishable corresponds to the hypothesis that

$$H_0: \beta_{1,m|n} = \dots = \beta_{K,m|n} = 0$$

which can be tested with either a Wald or an LR test. If alternatives are indistinguishable with respect to the variables in the model, then you can obtain more efficient estimates by combining them. Note, however, that while the `mlogtest` command makes it easier to test the hypotheses that each pair of outcomes can be combined, we do not recommend combining categories simply because the null hypothesis is not rejected. This is likely to lead to over-fitting your data and creating outcome variables that do not make substantive sense. Instead, these tests should be used to test a substantively motivated hypothesis that two categories are indistinguishable.

Wald test for combining alternatives

The command `mlogtest`, `combine` computes Wald tests of the null hypothesis that two alternatives can be combined for all pairs of alternatives. For example,

```
. mlogit party age income i.black i.female i.educ, base(5) nolog
(output omitted)
. mlogtest, combine
```

Wald tests for combining alternatives (N=1382)

H0: All coefficients except intercepts associated with a given pair of alternatives are 0 (i.e., alternatives can be combined)

	chi2	df	P>chi2
StrDem & Dem	72.854	6	0.000
StrDem & Indep	40.334	6	0.000
StrDem & Rep	126.561	6	0.000
StrDem & StrRep	83.272	6	0.000
Dem & Indep	15.141	6	0.019
Dem & Rep	44.862	6	0.000
Dem & StrRep	56.580	6	0.000
Indep & Rep	49.879	6	0.000
Indep & StrRep	60.203	6	0.000
Rep & StrRep	22.286	6	0.001

From these results, we can reject the hypothesis that categories `StrDem` and `Dem` are indistinguishable. Indeed, our results indicate that all the categories are distinguishable.

(Advanced) Using test [outcome]

The `mlogtest`, `combine` command makes its computations by using the `test` command for multiple-equation models. Although most researchers will find that `mlogtest` is sufficient for their needs, there might be situations in which you want to conduct tests that are unique to your application. If so, this section provides some insights into how to use the `test` command to test hypotheses involving combining outcomes.

To test that `StrDem` is indistinguishable from the base outcome `StrRep`, type

```
. test [StrDem]
( 1) [StrDem]age = 0
( 2) [StrDem]income = 0
( 3) [StrDem]0b.black = 0
( 4) [StrDem]1.black = 0
( 5) [StrDem]0b.female = 0
( 6) [StrDem]1.female = 0
( 7) [StrDem]1b.educ = 0
( 8) [StrDem]2.educ = 0
( 9) [StrDem]3.educ = 0
    Constraint 3 dropped
    Constraint 5 dropped
    Constraint 7 dropped
      chi2( 6) =   83.27
    Prob > chi2 =   0.0000
```

The result matches the results from `mlogtest` in row `StrDem & StrRep`. The command `test [outcome]` indicates which equation is being referenced in multiple-equation commands. `mlogit` is a multiple-equation command with $J - 1$ equations that are named by the value label for the outcome categories. In the output above, constraints 3, 5, and 7 were dropped. These constraints correspond to the base categories for factor variables. For example, `0b.black` is the coefficient for the excluded base outcome, which by definition is 0.

The test is more complicated when neither outcome that is being considered is the base. For example, to test that m and n are indistinguishable when the base outcome b is neither m nor n , the hypothesis is

$$H_0: (\beta_{1,m|b} - \beta_{1,n|b}) = \dots = (\beta_{K,m|b} - \beta_{K,n|b}) = 0$$

That is, you want to test the difference between two sets of coefficients. This is done with `test [outcome1=outcome2]`. For example, to test whether `StrDem` and `Dem` can be combined, type


```

. test [StrDem=Dem]
( 1) [StrDem]age - [Dem]age = 0
( 2) [StrDem]income - [Dem]income = 0
( 3) [StrDem]0b.black - [Dem]0b.black = 0
( 4) [StrDem]1.black - [Dem]1.black = 0
( 5) [StrDem]0b.female - [Dem]0b.female = 0
( 6) [StrDem]1.female - [Dem]1.female = 0
( 7) [StrDem]1b.educ - [Dem]1b.educ = 0
( 8) [StrDem]2.educ - [Dem]2.educ = 0
( 9) [StrDem]3.educ - [Dem]3.educ = 0
Constraint 3 dropped
Constraint 5 dropped
Constraint 7 dropped
      chi2( 6) =    72.85
Prob > chi2 =    0.0000

```

The results are identical to those from `mlogtest`.

LR test for combining alternatives

An LR test of combining m and n can be computed by first fitting the full model with no constraints, with the resulting LR statistic $LR \chi^2_F$. Then, fit a restricted model M_R in which outcome m is used as the base category and all the coefficients except the constant in the equation for outcome n are constrained to 0, with the resulting test statistic $LR \chi^2_R$. The test statistic for the test of combining m and n is the difference $LR \chi^2_{RvsF} = LR \chi^2_F - LR \chi^2_R$, which is distributed as chi-squared with K degrees of freedom, where K is the number of regressors. The command `mlogtest`, `lrcombine` computes $J \times (J - 1)$ tests for all pairs of outcome categories. For example,

```

. mlogit party age income i.black i.female i.educ, base(5) nolog
(output omitted)
. mlogtest, lrcombine
LR tests for combining alternatives (N=1382)
Ho: All coefficients except intercepts associated with a given pair
of alternatives are 0 (i.e., alternatives can be collapsed)

```

	chi2	df	P>chi2
StrDem & Dem	80.893	6	0.000
StrDem & Indep	44.075	6	0.000
StrDem & Rep	198.758	6	0.000
StrDem & StrRep	141.446	6	0.000
Dem & Indep	15.753	6	0.015
Dem & Rep	61.899	6	0.000
Dem & StrRep	73.214	6	0.000
Indep & Rep	60.872	6	0.000
Indep & StrRep	78.673	6	0.000
Rep & StrRep	22.894	6	0.001

(Advanced) Using constraints with lrtest

The `mlogtest`, `lrcombine` command computes LR tests by using the powerful `constraint` command (see [R] [constraint](#)). Although most researchers are likely to find `mlogtest` sufficient for their needs, some might want to learn more about how constraints are specified when fitting models.

An LR test that categories are indistinguishable can be computed using the command `constraint`. To test whether `StrDem` and `StrRep` are indistinguishable, we start by fitting the full model and storing the results:

```
. mlogit party age income i.black i.female i.educ, base(5) nolog
(output omitted)
. estimates store full_model
```

Second, we define a constraint by using the command

```
. constraint define 999 [StrDem]
```

We arbitrarily chose number 999 to label the constraint. Any integer from 1 to 1,999 inclusive can be used. The expression `[StrDem]` indicates that all coefficients should be estimated except for those fixed by the constraint. Third, we refit the model with this constraint. The base category must be `StrRep` (category 5) so that the coefficients indicated by `[StrDem]` are comparisons of `StrDem` and `StrRep`:

```
. mlogit party age income i.black i.female i.educ,
> constraint(999) base(5) nolog
(output omitted)
. estimates store constraint999
```

The model is fit with the constraint imposed, and results are stored using the name `constraint999`. Comparing the full model to the constrained model,

```
. lrtest full_model constraint999
Likelihood-ratio test
(Assumption: constraint999 nested in full_model)
```

LR chi2(6) =	141.45
Prob > chi2 =	0.0000

The result matches that from `mlogtest`, `lrcombine`.

8.4 Independence of irrelevant alternatives

The MNLM, as well as the conditional logit and rank-ordered logit models discussed below, make the assumption known as the independence of irrelevant alternatives (IIA). Here we describe the assumption in terms of the MNLM. In this model,

$$\frac{\Pr(y = m | \mathbf{x})}{\Pr(y = n | \mathbf{x})} = \exp \left\{ \mathbf{x} (\beta_{m|b} - \beta_{n|b}) \right\}$$

where the odds do not depend on other alternatives that are available. In this sense, those alternatives are "irrelevant". What this means is that adding or deleting alternatives does not affect the odds among the remaining alternatives.

This point is often made with the red bus–blue bus example. Suppose people in a city have three ways of getting to work: by car, by taking a bus operated by a company that uses red buses, or by taking a bus operated by an identical company that uses blue buses. We might expect that many people have a clear preference between taking the car versus taking a bus but are indifferent about whether they take a red bus or a blue bus. Suppose the odds of a person taking a red bus compared with those of taking a car are 1:1. IIA implies the odds will remain 1:1 between these two alternatives even if the blue bus company were to go out of business. The assumption is dubious because we would expect the vast majority of those who take the blue bus to have the red bus as their next preference. Consequently, eliminating the blue bus will increase the probability of traveling by red bus much more than it will increase the probability of someone traveling by car, yielding odds more like 2:1 than 1:1. In other words, because the blue bus and red bus are close substitutes, having the blue bus as an available alternative leads the MNLM to underestimate the preference for red bus versus car.

Tests of IIA involve comparing the estimated coefficients from the full model to those from a restricted model that excludes at least one of the alternatives. If the test statistic is significant, the assumption of IIA is rejected, indicating that the MNLM is inappropriate. In this section, we consider the two most common tests of IIA: the Hausman–McFadden (HM) test (Hausman and McFadden 1984) and the Small–Hsiao (SH) test (Small and Hsiao 1985). For details on other tests, see Fry and Harris (1996, 1998). For a model with J alternatives, we consider J ways of computing each test. If you remove the first alternative and refit the model, you get the first restricted model leading to the first variation of the test. If you remove the second alternative, you get the second variation, and so on. Each restricted model will lead to a different test statistic, as we demonstrate below.

Both the HM test and the SH test are computed by `mlogtest`, and for both tests we compute J variations. As many users of `mlogtest` have told us, the HM and SH tests often provide conflicting information on whether IIA has been violated, with some of the tests rejecting the null hypothesis, while others do not. To explore this further, Cheng and Long (2007) ran Monte Carlo experiments to examine the properties of these tests. Their results show that the HM test has poor size properties even with sample sizes of more than 1,000. For some data structures, the SH test has reasonable size properties for samples of 500 or more. But with other data structures, the size properties

are extremely poor and do not get better as the sample size increases. Overall, they conclude that these tests are not useful for assessing violations of the IIA property.

It appears that the best advice regarding IIA goes back to an early statement by McFadden (1974), who wrote that the multinomial and conditional logit models should be used only in cases where the alternatives “can plausibly be assumed to be distinct and weighted independently in the eyes of each decision maker”. Similarly, Amemiya (1981) suggests that the MNLM works well when the alternatives are dissimilar. Care in specifying the model to involve distinct alternatives that are not substitutes for one another seems to be reasonable—albeit unfortunately ambiguous—advice.

Caution regarding tests of IIA. We do not believe that tests of IIA are useful, but we have heard from readers about reviewers or editors who insist that they provide the results of an IIA test. In our experience, you can almost always obtain some tests that accept the null and others that reject the null when using the same model with the same data. We would try to convince those requesting the test that these tests do not provide useful information, perhaps citing our book, along with Fry and Harris (1996, 1998) and Cheng and Long (2007). If this does not work, you may still need to provide the test results. In this section, we tell you how to compute them and illustrate their limitations.

8.4.1 Hausman–McFadden test of IIA

The HM test of IIA involves the following steps:

1. Fit the full model with all J alternatives included, with estimates in $\hat{\beta}_F$.
2. Fit a restricted model by eliminating one or more alternatives, with estimates in $\hat{\beta}_R$.
3. Let $\hat{\beta}_F^*$ be a subset of $\hat{\beta}_F$ after eliminating coefficients not fit in the restricted model. The test statistic is

$$HM = (\hat{\beta}_R - \hat{\beta}_F^*)' \left\{ \widehat{\text{Var}}(\hat{\beta}_R) - \widehat{\text{Var}}(\hat{\beta}_F^*) \right\}^{-1} (\hat{\beta}_R - \hat{\beta}_F^*)$$

where HM is asymptotically distributed as chi-squared with degrees of freedom equal to the rows in $\hat{\beta}_R$ if IIA is true. Significant values of HM indicate that the IIA assumption has been violated.

The HM test of IIA can be computed with `mlogtest`:

```
. mlogit party age income i.black i.female i.educ, base(5)
(output omitted)
. mlogtest, hausman
Hausman tests of IIA assumption (N=1382)
Ho: Odds(Outcome-J vs Outcome-K) are independent of other alternatives
```

	chi2	df	P>chi2
StrDem	4.622	20	1.000
Dem	0.919	21	1.000
Indep	-2.244	19	.
Rep	3.030	21	1.000
StrRep	-0.580	21	.

Note: A significant test is evidence against Ho.
 Note: If chi2<0, the estimated model does not meet asymptotic assumptions.

Five tests of IIA are reported. The first four correspond to excluding one of the four non-base categories. The fifth test, in row **StrRep**, is computed by refitting the model with the largest remaining outcome as the base category.³ Three of the tests produce negative chi-squareds, something that is common with this test. Hausman and McFadden (1984, 1226) note this possibility and conclude that a negative result is evidence that IIA has not been violated. Our simulations suggest that negative chi-squareds indicate problems with the test, consistent with the warning that the `mlogtest` output provides.

8.4.2 Small-Hsiao test of IIA

To compute an SH test, the sample is divided randomly into two subsamples of about equal size. The unrestricted MNLM is fit on both subsamples, where $\hat{\beta}_u^{S_1}$ contains estimates from the unrestricted model on the first subsample and $\hat{\beta}_u^{S_2}$ is its counterpart for the second subsample. A weighted average of the coefficients is computed as

$$\hat{\beta}_u^{S_1 S_2} = \left(\frac{1}{\sqrt{2}} \right) \hat{\beta}_u^{S_1} + \left\{ 1 - \left(\frac{1}{\sqrt{2}} \right) \right\} \hat{\beta}_u^{S_2}$$

Next, a restricted sample is created from the second subsample by eliminating all cases with a chosen value of the dependent variable. The MNLM is fit using the restricted sample, yielding the estimates $\hat{\beta}_r^{S_2}$ and the likelihood $L(\hat{\beta}_r^{S_2})$. The SH statistic is

$$SH = -2 \left\{ L \left(\hat{\beta}_u^{S_1 S_2} \right) - L \left(\hat{\beta}_r^{S_2} \right) \right\}$$

which is asymptotically distributed as chi-squared with degrees of freedom equal to the number of coefficients that are fit in both the full model and the restricted model.

3. Even though `mlogtest` fits additional models to compute various tests, when the command ends, it restores the estimates from your original model. Consequently, commands that require results from your original `mlogit`, such as `predict` and `m*` commands, will work correctly.

To compute the SH test, use the command `mlogtest`, `smhsiao` (our program uses code from `smhsiao` by Nick Winter [2000] available at the Statistical Software Components archive). Because the SH test requires randomly dividing the data into subsamples, the results will differ with successive calls of the command, because the sample will be randomly divided differently. To obtain test results that can be replicated, you must explicitly set the seed used by the random-number generator. For example,

```
. set seed 124386
. mlogtest, smhsiao
Small-Hsiao tests of IIA assumption (N=1382)
Ho: Odds(Outcome-J vs Outcome-K) are independent of other alternatives
```

	lnL(full)	lnL(omit)	chi2	df	P>chi2
StrDem	-696.753	-690.654	12.198	21	0.934
Dem	-565.571	-557.488	16.166	21	0.760
Indep	-764.563	-758.290	12.547	21	0.924
Rep	-621.562	-615.492	12.140	21	0.936
StrRep	-761.598	-752.804	17.587	21	0.675

Note: A significant test is evidence against Ho.

These results are consistent with those from the HM test, with none of the tests being significant.

Before taking these results seriously, we tried three other seeds to produce a different random division of the sample. The results varied widely. For example,

```
. set seed 254331
. mlogtest, smhsiao
Small-Hsiao tests of IIA assumption (N=1382)
Ho: Odds(Outcome-J vs Outcome-K) are independent of other alternatives
```

	lnL(full)	lnL(omit)	chi2	df	P>chi2
StrDem	-727.367	-692.048	70.639	21	0.000
Dem	-610.636	-573.268	74.736	21	0.000
Indep	-783.456	-747.654	71.604	21	0.000
Rep	-650.962	-615.434	71.057	21	0.000
StrRep	-751.887	-740.193	23.388	21	0.324

Note: A significant test is evidence against Ho.

Using the new seed, we reject the null at the 0.001 level in four of the five tests, illustrating a common problem when using the SH test—you often get very different results depending on how the sample is randomly divided.

Tip: Setting the random seed. The random numbers that divide the sample for the SH test are based on the `runiform()` function, which uses a pseudorandom-number generator to create a sequence of numbers based on a seed number. Although these numbers appear to be random, the same sequence will be generated each time you start with the same seed. In this sense (and some others), these numbers are pseudorandom rather than random. If you specify the seed with `set seed #`, you ensure that you can replicate your results later. See [R] `set seed` for more details.

8.5 Measures of fit

As with models for binary and ordinal outcomes, many scalar measures of fit for the MNLM model can be computed with the `SPost` command `fitstat`, and information criteria can be computed with `estat ic`. The same caveats against overstating the importance of these scalar measures apply here as to the other models we have considered (see also chapter 3). To examine the fit of individual observations, you can fit the series of binary logits implied by the MNLM and use the established methods of examining the fit of observations to binary logit estimates.

8.6 Overview of interpretation

Although the MNLM is a mathematically simple extension of the binary model, interpretation is difficult because of the many possible comparisons. Even in our simple example with five outcomes, we have 10 comparisons: `StrDem` versus `StrRep`, `Dem` versus `StrRep`, `Indep` versus `StrRep`, `Rep` versus `StrRep`, `StrDem` versus `Rep`, `Dem` versus `Rep`, `Indep` versus `Rep`, `StrDem` versus `Indep`, `Dem` versus `Indep`, and `StrDem` versus `Dem`. It is tedious to write all of them, let alone to interpret all of them for each independent variable. The key to effective interpretation is to avoid overwhelming yourself or your audience with the many comparisons.

As with models for binary and ordinal outcomes, we prefer methods of interpretation that are based on predicted probabilities. Fortunately, these methods are essentially unchanged from those used for ordinal models in the last chapter, where the predicted probability is now computed with the formula

$$\widehat{\Pr}(y = m \mid \mathbf{x}) = \frac{\exp(\mathbf{x}\hat{\beta}_{m|J})}{\sum_{j=1}^J \exp(\mathbf{x}\hat{\beta}_{j|J})} \quad (8.2)$$

where \mathbf{x} can contain either hypothetical values or values based on cases in the sample. Here we assume that the base outcome is J , but any base could be used.

We follow a similar order of presentation to that used in the last chapter, providing new variations in some cases and excluding some topics. In all cases, however, methods

from chapter 7 could be used for nominal models, and new ideas shown in this chapter could be applied to ordinal models. For example, while we do not consider ideal types in this chapter, they are just as useful for nominal outcomes as they were for the ordinal regression model (ORM).

We begin by examining the distribution of predictions for each observation in the estimation sample. Then, we consider how marginal effects can be used as an overall assessment of the impact of each variable, but we also show what can be learned by looking at the distribution of effects for each observation in the estimation sample. We extend earlier methods for examining tables of predictions and show how to test a difference of differences. Next, we plot predictions as a continuous independent variable changes, which we use to highlight how results from an ordinal model can be misleading when an outcome does not behave as if it were ordinal. Finally, we consider interpretation using odds ratios. Although odds ratios in the MNLM have all the limitations discussed in chapter 6, they are important for understanding how independent variables affect the distribution of observations between pairs of outcomes, something that cannot be done using predicted probabilities alone.

Before beginning, we must also emphasize once again that, as with other models considered in this book, the MNLM is nonlinear in the outcome probabilities, and no approach can fully describe the relationship between an independent variable and the outcome probabilities. You should experiment with each of these methods before deciding which approach is most effective in your application.

8.7 Predicted probabilities with predict

The most basic command for computing probabilities is `predict`. After fitting the model with `mlogit`, predicted probabilities for all outcomes within the sample can be calculated with

```
predict newvarlist [if] [in]
```

where you must provide one new variable name for each of the J categories of the dependent variable, ordered from the lowest to the highest numerical values. For example,

```
. mlogit party age income i.black i.female i.educ, base(5)
(output omitted)
. estimates store mlogit
. predict mnlnSD mnlnD mnlnI mnlnR mnlnSR
(option pr assumed; predicted probabilities)
```


The variables created by `predict` are

```
. codebook mnlmSD mnlmD mnlmI mnlmR mnlmSR, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
mnlmSD	1382	1193	.1924747	.0212015	.7654322	Pr(party==StrDem)
mnlmD	1382	1193	.3089725	.130991	.5125323	Pr(party==Dem)
mnlmI	1382	1193	.1092619	.0266534	.2838254	Pr(party==Indep)
mnlmR	1382	1193	.2670043	.0141873	.5036324	Pr(party==Rep)
mnlmSR	1382	1193	.1222865	.0047189	.4662779	Pr(party==StrRep)

As with the ordinal model, if you specify a single variable name after `predict`, you will obtain predicted probabilities for one outcome category, which you can specify using the `outcome()` option.

As discussed in section 7.10, examining the distribution of the in-sample predictions can be used to get a general sense of what is going on in your model and can sometimes uncover problems in your data. The distribution of predictions can also be used to informally compare competing models, which we illustrate next.

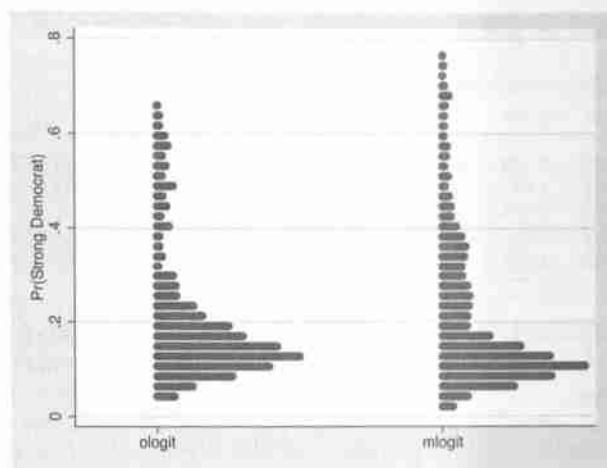
We could reasonably argue that the five categories of our dependent variable `party` are an ordinal scale of party affiliation. Accordingly, it seems reasonable to model these data with an ordinal logit model. First, we fit the model and compute predictions:

```
. ologit party age income i.black i.female i.educ
(output omitted)
. predict olmSD olmD olmI olmR olmSR
(option pr assumed; predicted probabilities)
. codebook olm*, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
olmSD	1382	1193	.1934016	.042849	.6546781	Pr(party==1)
olmD	1382	1193	.30611	.1393038	.3808963	Pr(party==2)
olmI	1382	1193	.1091385	.0349378	.1221904	Pr(party==3)
olmR	1382	1193	.269342	.0484485	.3877065	Pr(party==4)
olmSR	1382	1193	.1220079	.0124719	.3484676	Pr(party==5)

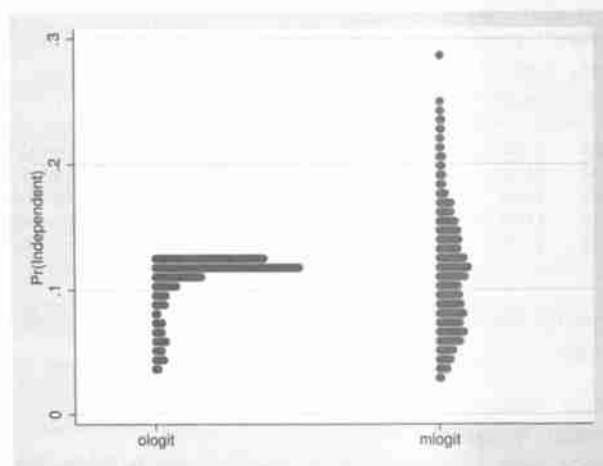
Next, we plot the predicted probability of being a strong Democrat (outcome 1) with the `dotplot` command:

```
. label var olmSD "ologit"
. label var mnlmSD "mlogit"
. dotplot olmSD mnlmSD, ylabel(0(.2).8, grid) ytitle(Pr(Strong Democrat))
```

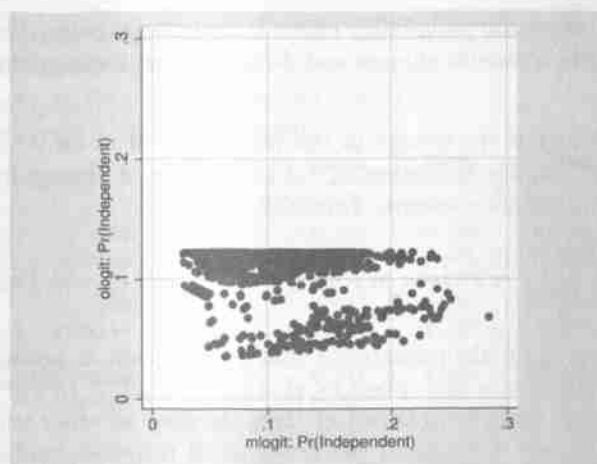



The histograms are similar, and the correlation between the predictions for the ordered logit model (OLM) and the MNLM is 0.94. This suggests that the conclusions from the two models might be similar.

If we look at the middle category of Independent, however, things look quite different, reflecting the abrupt truncation of the distribution of predictions for middle categories that is often found with the OLM:



Not only are the distributions quite different in shape, but the correlation between the predicted probabilities is negative: -0.19 ! A scatterplot of the predictions shows striking differences between the two models:



A low correlation between the predictions from the MNLM and the ORM could reflect a lack of ordinality, but this is not necessarily so. For example, in simulations where data were generated to meet the assumptions of the ORM, we found low correlations in predictions between middle categories between the MNLM and the ORM when the size of the assumed error variance in the ORM was large relative to the size of the regression coefficients. Nonetheless, when predictions are very different between an ordinal and nominal regression model, we recommend considering the appropriateness of the ordinal model. This is considered further in section 8.10, where we plot predictions from the MNLM and the OLM.

8.8 Marginal effects

Average marginal effects are a quick and valuable way to assess the effects of all the independent variables in your model. Because methods for using marginal effects to interpret the MNLM are identical to those used for the OLM in section 7.11, we only review key points here. We then extend materials from chapter 7 by examining the distribution of effects within the estimation sample. Marginal effects are also used in section 8.11.2 when we plot odds ratios.

The marginal change is the slope of the curve relating x_k to $\Pr(y=m|\mathbf{x})$, holding all other variables constant, where $\Pr(y=m|\mathbf{x})$ is defined by (8.2). For the MNLM, the marginal change is

$$\frac{\partial \Pr(y=m|\mathbf{x})}{\partial x_k} = \Pr(y=m|\mathbf{x}) \left\{ \beta_{k,m|J} - \sum_{j=1}^J \beta_{k,j|J} \Pr(y=j|\mathbf{x}) \right\}$$

Because this equation combines all the $\beta_{k,j|J}$'s, the value of the marginal change depends on the levels of all variables in the model and can switch sign at different values of these variables. Also, because the marginal change is the instantaneous rate or change, it

can be misleading when the probability curve is changing rapidly. For this reason, we generally prefer using a discrete change and do not discuss the marginal change further in this chapter.

The discrete change is the change in the probability of m for a change in x_k from the start value x_k^{start} to the end value x_k^{end} (for example, a change from $x_k^{\text{start}} = 0$ to $x_k^{\text{end}} = 1$), holding other x 's constant. Formally,

$$\frac{\Delta \Pr(y = m \mid \mathbf{x})}{\Delta x_k (x_k^{\text{start}} \rightarrow x_k^{\text{end}})} = \Pr(y = m \mid \mathbf{x}, x_k = x_k^{\text{end}}) - \Pr(y = m \mid \mathbf{x}, x_k = x_k^{\text{start}})$$

where $\Pr(y = m \mid \mathbf{x}, x_k)$ is the probability that $y = m$ given \mathbf{x} , noting a specific value for x_k . The change indicates that when x_k changes from x_k^{start} to x_k^{end} , the probability of outcome m changes by $\Delta \Pr(y = m \mid \mathbf{x}) / \Delta x_k$, holding all other variables at \mathbf{x} . The magnitude of the change depends on the levels of all variables, including x_k , and the size of the change in x_k that is being evaluated.

Marginal effects are computed by `mchange` as discussed in detail in sections 4.5.4 and 7.11. After fitting the model used as our running example, we compute the average discrete changes for a standard deviation change in continuous variables and a change from 0 to 1 for other variables. The option `amount(sd)` suppresses the default computation of marginal changes and discrete changes of 1 unit, while `width(8)` prevents results wrapping due to the long value labels for `educ`:


```
. estimates restore mlogit
. mchange, amount(sd) brief width(8)
mlogit: Changes in Pr(y) | Number of obs = 1382
Expression: Pr(party), predict(outcome())
```

		StrDem	Dem	Indep	Rep	StrRep
age						
	+SD	0.054	-0.033	-0.024	-0.030	0.032
	p-value	0.000	0.006	0.001	0.009	0.003
income						
	+SD	-0.039	-0.022	-0.003	0.041	0.023
	p-value	0.001	0.122	0.752	0.002	0.019
black						
	yes vs no	0.274	0.047	0.041	-0.248	-0.113
	p-value	0.000	0.220	0.142	0.000	0.000
female						
	yes vs no	-0.006	0.065	-0.024	-0.004	-0.031
	p-value	0.768	0.010	0.153	0.856	0.078
educ						
	hs only vs not hs grad	-0.045	0.031	-0.039	0.027	0.026
	p-value	0.137	0.414	0.208	0.466	0.254
	college vs not hs grad	-0.083	0.041	-0.092	0.034	0.100
	p-value	0.025	0.367	0.007	0.441	0.001
	college vs hs only	-0.037	0.010	-0.052	0.006	0.073
	p-value	0.142	0.744	0.004	0.825	0.002

Even for this relatively simple model and looking only at a single amount of change for each variable, there is a lot of information to digest. To make it simpler to interpret these results, we plot the changes with `mchangeplot` (see `help mchangeplot` and section 6.2 for additional information about `mchangeplot`). We begin by looking at the average discrete changes for standard deviation increases in age and income:

```
. mchangeplot age income,
> symbols(D d i r R) min(-.05) max(.05) gap(.02) sig(.05)
> xtitle(Average Discrete Change) ysize(1.3) scale(2.1)
```

By default, `mchangeplot` represents each outcome category with the first letter of the value label for that category. In this example, this would be confusing because the categories `StrDem` and `StrRep` both begin with S. The `symbols()` option lets you specify one or more letters for each category. For example, we could use `symbol(SD D I R SR)`. Or, as we prefer, we can use `symbol(D d i r R)` so that capitals indicate more strongly held affiliations. The resulting graph looks like this, where the *'s indicate that an effect is significant at the 0.05 level:

can be misleading when the probability curve is changing rapidly. For this reason, we generally prefer using a discrete change and do not discuss the marginal change further in this chapter.

The discrete change is the change in the probability of m for a change in x_k from the start value x_k^{start} to the end value x_k^{end} (for example, a change from $x_k^{\text{start}} = 0$ to $x_k^{\text{end}} = 1$), holding other x 's constant. Formally,

$$\frac{\Delta \Pr(y = m \mid \mathbf{x})}{\Delta x_k (x_k^{\text{start}} \rightarrow x_k^{\text{end}})} = \Pr(y = m \mid \mathbf{x}, x_k = x_k^{\text{end}}) - \Pr(y = m \mid \mathbf{x}, x_k = x_k^{\text{start}})$$

where $\Pr(y = m \mid \mathbf{x}, x_k)$ is the probability that $y = m$ given \mathbf{x} , noting a specific value for x_k . The change indicates that when x_k changes from x_k^{start} to x_k^{end} , the probability of outcome m changes by $\Delta \Pr(y = m \mid \mathbf{x}) / \Delta x_k$, holding all other variables at \mathbf{x} . The magnitude of the change depends on the levels of all variables, including x_k , and the size of the change in x_k that is being evaluated.

Marginal effects are computed by `mchange` as discussed in detail in sections 4.5.4 and 7.11. After fitting the model used as our running example, we compute the average discrete changes for a standard deviation change in continuous variables and a change from 0 to 1 for other variables. The option `amount(sd)` suppresses the default computation of marginal changes and discrete changes of 1 unit, while `width(8)` prevents results wrapping due to the long value labels for `educ`:

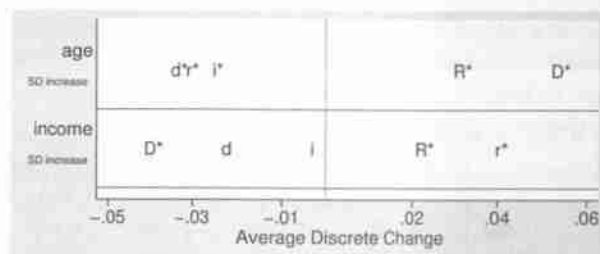

```
. estimates restore mlogit
. mchange, amount(sd) brief width(8)
mlogit: Changes in Pr(y) | Number of obs = 1382
Expression: Pr(party), predict(outcome())
```

		StrDem	Dem	Indep	Rep	StrRep
age						
	+SD	0.054	-0.033	-0.024	-0.030	0.032
	p-value	0.000	0.006	0.001	0.009	0.003
income						
	+SD	-0.039	-0.022	-0.003	0.041	0.023
	p-value	0.001	0.122	0.752	0.002	0.019
black						
	yes vs no	0.274	0.047	0.041	-0.248	-0.113
	p-value	0.000	0.220	0.142	0.000	0.000
female						
	yes vs no	-0.006	0.065	-0.024	-0.004	-0.031
	p-value	0.768	0.010	0.153	0.856	0.078
educ						
	hs only vs not hs grad	-0.045	0.031	-0.039	0.027	0.026
	p-value	0.137	0.414	0.208	0.466	0.254
	college vs not hs grad	-0.083	0.041	-0.092	0.034	0.100
	p-value	0.025	0.367	0.007	0.441	0.001
	college vs hs only	-0.037	0.010	-0.052	0.006	0.073
	p-value	0.142	0.744	0.004	0.825	0.002

Even for this relatively simple model and looking only at a single amount of change for each variable, there is a lot of information to digest. To make it simpler to interpret these results, we plot the changes with `mchangeplot` (see `help mchangeplot` and section 6.2 for additional information about `mchangeplot`). We begin by looking at the average discrete changes for standard deviation increases in age and income:

```
. mchangeplot age income,
> symbols(D d i r R) min(-.05) max(.05) gap(.02) sig(.05)
> xtitle(Average Discrete Change) ysize(1.3) scale(2.1)
```

By default, `mchangeplot` represents each outcome category with the first letter of the value label for that category. In this example, this would be confusing because the categories `StrDem` and `StrRep` both begin with S. The `symbols()` option lets you specify one or more letters for each category. For example, we could use `symbol(SD D I R SR)`. Or, as we prefer, we can use `symbol(D d i r R)` so that capitals indicate more strongly held affiliations. The resulting graph looks like this, where the *'s indicate that an effect is significant at the 0.05 level:



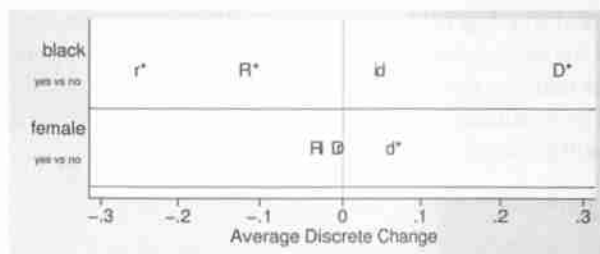
Before proceeding, you should verify that this graph corresponds to the output above from `mchange`. Although we probably would not include this graph in a paper, we use it to help describe the effects:

On average, a standard deviation increase in age, about 17 years, increases the probability of being a strong Republican by 0.03 and of being a strong Democrat by 0.05. The probabilities of other affiliations all decrease by roughly 0.03. All effects are significant at the 0.01 level.

On average, a standard deviation increase in income, roughly \$28,000, significantly increases the probability of being a Republican by 0.04 and a strong Republican by 0.02, while significantly decreasing the probability of being a strong Democrat by 0.04.

The greater effect of race compared with gender on party affiliation is shown with a plot of their average discrete changes. We use the following command to create the graph:

```
. mchangeplot black female,
> symbols(D d i r R) min(-.3) max(.3) gap(.1) sig(.05)
> xtitle(Average Discrete Change) ysize(1.3) scale(2.1)
```

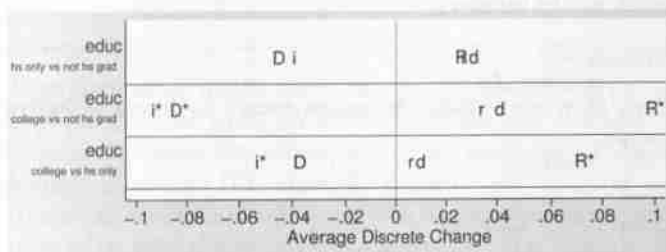


We conclude the following:

On average, for people similar on other characteristics, being black increases the probability of being a strong Democrat by nearly 0.30 compared with someone who is white. Conversely, being black decreases the probability of being a strong Republican by 0.11 and being a Republican by 0.25. Except for an increase of 0.07 in Democratic affiliation, the effects of gender are not significant.

For the factor variable `educ` with three categories, `mchange` provides all the pairwise contrasts, comparing those who have a high school diploma with those who do not, those who have a college degree with those who do not have a high school diploma, and those who have a college degree with those who have a high school diploma. One contrast is redundant in the sense that it can be computed from the other two. Still, it is useful to examine all contrasts to find patterns.

```
. mchangeplot educ,
> symbols(D d i r R) min(-.1) max(.1) gap(.02) sig(.05)
> xtitle(Average Discrete Change) ysize(1.3) scale(2.1)
```



We conclude the following:

Higher education increases the probability of identifying as a strong Republican and decreases the probability of identifying as a strong Democrat. Based on the distribution of other characteristics in the population, if we compare those who do not have a high school diploma with those who have graduated from college, the probability of being a strong Republican increases by 0.10 on average and the probability of being a strong Democrat decreases by 0.08.

Although we do not illustrate these methods here, we could examine other amounts of change and customize the plots with the `mchangeplot` options described in `help mchangeplot`.

8.8.1 (Advanced) The distribution of marginal effects

Because the average marginal effect (AME) is an average, it does not indicate variation in the sample. We find examining the distribution of marginal effects is often very useful, but it requires using loops, macros, and returns and is computationally intensive. After you are familiar with marginal effects and Stata tools for automation, we encourage you to study this section carefully. Initially, however, we hope you will at least skim it.

The value of a marginal effect depends on the level of all variables in the model (see section 6.2.5). Neither the AME nor the marginal effect at the mean provide information about how much variation there is within the sample in the size of the effects. In this section, we extend the methods from chapter 6 to the MNLM. The same techniques can also be used for the OLM.

The following commands generate the variable `incomedc` containing the discrete change for a standard deviation increase in `income`, where we assume that the estimation results from `mlogit` are in memory:

```

1] gen incomedc = .
2] label var incomedc ///
   "Effect of a one standard deviation change in income on Pr(Dem)"
3] sum income
4] local sd = r(sd)
5] local nobs = _N
6] forvalues i = 1/'nobs' {
7]     quietly {
8]         margins in `i', nose predict(outcome(2)) /// Dem
           at(income=gen(income)) at(income=gen(income+'sd'))
9]         local prstart = e1(r(b),1,1)
10]        local premd = e1(r(b),1,2)
11]        local dc = `premd' - `prstart'
12]        replace incomedc = `dc' in `i'
13]    }
14] }
```

Lines 1 and 2 create and label the variable that will hold the discrete change for each observation. Because we want to compute the effect of a standard deviation change in `income`, lines 3 and 4 compute the standard deviation and create the local macro `sd` containing the standard deviation. This is used in the `margins` command in line 8. Line 5 creates the macro `nobs` with the number of observations, which we use in line 6 to begin a `forvalues` loop through the 1,382 observations in the estimation sample.

The loop over observations is defined in lines 6 through 14. Because we do not want to see the output from `margins`, line 7 uses `quietly` to suppress the output. Line 8 uses `margins` to compute predictions for observation ``i'` for the second outcome category,

where `nose` suppresses the computation of the standard error (this speeds up the computations). The first `at()` statement specifies the observed value of `income`, with all other variables held at their observed values; the second `at()` specifies the prediction at one standard deviation more than the observed value. `margins` returns the predictions to the matrix `r(b)`, and lines 9 and 10 retrieve the starting and ending probabilities. Line 11 computes the discrete change. Line 12 saves the effect for observation ``i'` of variable `incomedc`. The last two lines terminate the `quietly` command and the `forvalues` loop.

Why computing the distribution of effects is slow. Computing effects for individual observations is slow. Every time `margins` is run, it computes predictions for all observations in the sample before it computes predictions at values specified by `at()`. In line 8, we need the predictions for a single observation, but `margins` computes predictions for all observations. You cannot turn off this behavior. Unfortunately, `margins` does not save the predictions it computes for each observation. If it did, we would not need the loop! Accordingly, for each observation in our loop, `margins` is computing $2N + 2$ predictions, for a total of $2(N^2 + N)$ predictions—nearly 4 million in our example! Using Stata/MP for eight cores, our loop took 60 seconds to complete. But we believe it is worth the time.

Although these commands might seem complex at first, the good news is that you can easily modify our code to work for other variables (for example, change `income` to `age`), for different outcomes (for example, change `outcome(2)` to `outcome(5)`), or for different amounts of change (for example, change ``sd'` to 1 for a discrete change of 1). Further, the same commands will work with models for binary, ordinal, and count outcomes or, indeed, for almost any model supported by the `margins` command.

To plot the distribution of effects, we first compute the mean of `incomedc` and assign it to a local macro named `ame`:

```
. sum incomedc
```

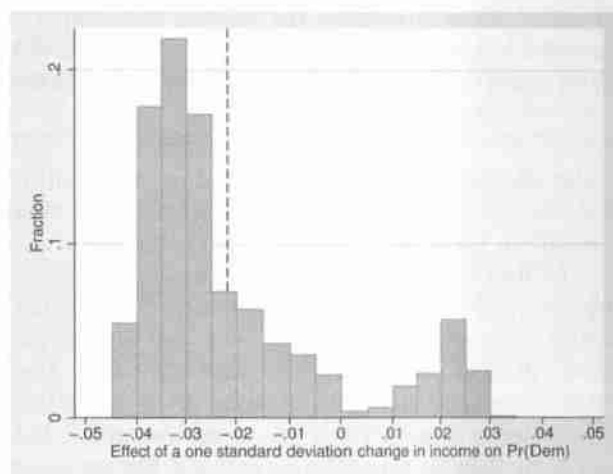
Variable	Obs	Mean	Std. Dev.	Min	Max
incomedc	1382	-.0217109	.0190934	-.0448823	.0305349

```
. local ame = r(mean)
```

The mean equals the AME of -0.022 computed by `mchange` on page 417. Next, we use `histogram` to plot the distribution of effects, with a vertical dashed line showing the value of the AME:

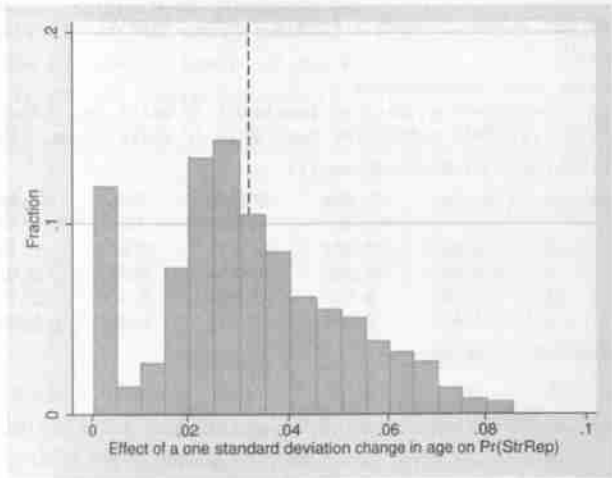
```
. histogram incomedc, xlabel(-.05(.01).05) fraction
> width(.005) color(gs10) fcolor(gs12) ylab(0(.1).2)
> xline(`ame', lpat(dash))
(bin=16, start=-.04455748, width=.005)
```


The histogram shows the distribution of discrete changes in the probability of being Democrat for a standard deviation change in income with the AME represented by a vertical, dashed line:



Even the sign of the AME is potentially misleading. The distribution of effects is bimodal with most of the sample having effects around -0.03 and a smaller group with positive effects near 0.025 . Suppose the independent variable being considered is an intervention where the spikes corresponded to two groups—say, whites and blacks—with negative effects for the larger group and positive effects for the smaller group. If substantive interest was on how the intervention would affect the smaller group, the AME is misleading because it is dominated by the negative effects for the larger group.

As a second example, we compute (without showing the commands) the distribution of the effects of `age` on being a strong Republican:



The AME of age on being a strong Republican is 0.030. The distribution of discrete changes ranges from 0.0004 to 0.082, with a spike near 0 followed by a gap until around 0.01. If age was a policy-relevant variable and the focus of the intervention was on individuals who had marginal effects near 0, the AME would be quite misleading.

8.9 Tables of predicted probabilities

As with models for binary and ordinal outcomes, tables of predictions can provide useful insights into models when there are substantively important, categorical independent variables. Because exactly the same commands can be used with nominal models as with ordinal models, we do not repeat the types of examples shown before (see section 7.13 for details). Instead, we focus on using tables of probabilities to compare and elaborate discrete changes across groups.

To show the effects of race and gender on party affiliation, we use `mtable` to compute probabilities for each combination of race and gender, holding other variables at their means. Although we could use the specification `at(black=(0 1) female=(0 1))`, we instead use multiple `at()` options to arrange the output in the order we prefer:


```
. mlogit party age income i.black i.female i.educ, base(5)
(output omitted)
. mtable, atmeans noci norownumbers
> at(black=0 female=0) at(black=1 female=0) // white men, black men
> at(black=0 female=1) at(black=1 female=1) // white women, black women
```

Expression: Pr(party), predict(outcome())

black	female	StrDem	Dem	Indep	Rep	StrRep
0	0	0.142	0.287	0.115	0.311	0.145
1	0	0.440	0.328	0.162	0.049	0.021
0	1	0.138	0.354	0.092	0.304	0.111
1	1	0.417	0.394	0.127	0.047	0.015

Specified values of covariates

	age	income	2. educ	3. educ
Current	45.9	37.5	.58	.259

We can interpret this as follows:

For those who are average on all other characteristics, blacks are far more likely than whites to be strong Democrats and far less likely to be Republicans or strong Republicans. Much smaller differences are found between men and women in party affiliation.

Supposing our substantive interest focuses on race and gender differences in party affiliation, we would want to test the differences in predictions between groups. The easiest way to do this is with `mchange`, using the option `statistics(start end change pvalue)` to list the predicted probabilities for each group as well as the discrete changes. The effects of race for women (`at(female=1)`) are


```
. mchange black, at(female=1) atmeans brief
>      statistics(start end change pvalue) title(Effect of race for women)
Effect of race for women | Number of obs = 1382
Expression: Pr(party), predict(outcome())
```

		StrDem	Dem	Indep	Rep	StrRep
black						
	From	0.138	0.354	0.092	0.304	0.111
	To	0.417	0.394	0.127	0.047	0.015
	yes vs no	0.278	0.040	0.035	-0.257	-0.096
	p-value	0.000	0.342	0.169	0.000	0.000

We interpret this as follows:

Compared with a white woman who is average on all characteristics, an otherwise similar black woman has a 0.28 higher probability of being a strong Democrat, a 0.26 lower probability of being a Republican, and a 0.10 lower probability of being a strong Republican. All differences are significant at the 0.001 level.

Similarly, we compute the effects for men:

```
. mchange black, at(female=0) atmeans brief
>      statistics(start end change pvalue) title(Effect of race for men)
Effect of race for men | Number of obs = 1382
Expression: Pr(party), predict(outcome())
```

		StrDem	Dem	Indep	Rep	StrRep
black						
	From	0.142	0.287	0.115	0.311	0.145
	To	0.440	0.328	0.162	0.049	0.021
	yes vs no	0.298	0.041	0.047	-0.262	-0.125
	p-value	0.000	0.293	0.126	0.000	0.000

Although the effects for race are of roughly the same size for men and women, we would like to test whether they are equal. For example, the discrete change for women is 0.278 and for men is 0.298. Can we say these differences are significantly different from one another? We consider this question in the next section.

8.9.1 (Advanced) Testing second differences

Computing and testing second differences is extremely useful, especially in group comparisons. To make these computations requires more advanced programming and a deeper understanding of how `margins` and `lincom` work. On first reading, you might want to only skim this section. However, we hope you return to it later.

Earlier, we used `mchange` to compute the first difference for race holding `female` at either 0 or 1 with other variables held at their means. Now, we want to test the null hypothesis that the discrete change for men is equal to the discrete change for women:

$$H_0: \frac{\Delta \Pr(y = j \mid \bar{x}, \text{female} = 0)}{\Delta \text{black}(0 \rightarrow 1)} = \frac{\Delta \Pr(y = j \mid \bar{x}, \text{female} = 1)}{\Delta \text{black}(0 \rightarrow 1)}$$

We begin by fitting the model and storing the estimates so that we can restore them after posting estimates from `margins`:

```
. mlogit party age income i.black i.female i.educ, base(5)
(output omitted)
. estimates store mymodel
```

Now, we use `margins` to compute predictions for all combinations of gender and race for outcome 1. Later, we will use a loop to make computations for all outcomes. Because the `atlegend` produced by `margins` is in this case quite long, we suppress it with the `noatlegend` option and use `mlstat` to list the values at which the independent variables are held:

```
. margins, predict(outcome(1)) post atmeans noatlegend
> at(black=0 female=0) at(black=1 female=0) // white men, black men
> at(black=0 female=1) at(black=1 female=1) // white women, black women

Adjusted predictions      Number of obs   =      1382
Model VCE      : OIM
Expression      : Pr(party==StrDem), predict(outcome(1))
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
_at						
1	.1423217	.0139387	10.21	0.000	.1150023	.1696411
2	.4404284	.0436357	10.09	0.000	.354904	.5259528
3	.1381515	.0141093	9.79	0.000	.1104977	.1658052
4	.4165218	.042888	9.71	0.000	.3324629	.5005807

```
. mlstat
at() values held constant
```

age	income	2. educ	3. educ
45.9	37.5	.58	.259

```
at() values vary
```

_at	black	female
1	0	0
2	1	0
3	0	1
4	1	1

The `post` option saves the predictions so they can be used with `lincom` or `mlincom` to compute second differences.

Why are we using `margins`, which computes predictions for only one outcome, instead of `mtable`, which computes predictions for all outcomes? To test predictions, those predictions must be saved to the `e(b)` and `e(V)` matrices. Because `margins` computes predictions for only one outcome at a time, we can only post predictions for one outcome. Although `mtable` can collect predictions for all the outcomes, it can only post predictions for a single outcome, just like `margins`. Accordingly, there is no advantage to using `mtable`.

The second difference is computed by taking the difference between two differences: 1) the difference between the probability for black men contained in `_b[2._at]` and for white men in `_b[1._at]`; and 2) the difference between the probability for black women in `_b[4._at]` and for white women in `_b[3._at]`:

```
. lincom (_b[2._at] - _b[1._at]) - (_b[4._at] - _b[3._at])
(1)  - 1bn._at + 2._at + 3._at - 4._at = 0
```

	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
(1)	.0197363	.0218253	0.90	0.366	-.0230405	.062513

```
. est restore mymodel
(results mymodel are active now)
```

The second difference for the first outcome, being a strong Democrat, is less than two points and not significantly different from 0. `estimates restore mymodel` restores the `mlogit` estimates to compute the second difference for other outcomes.

We can automate the process with a `forvalues` loop over the values of the outcome:

```
1] forvalues iout = 1/5 {
2]     margins, predict(outcome(`iout`)) post atmeans
        at(black=0 female=0) at(black=1 female=0) // white men, black men ///
        at(black=0 female=1) at(black=1 female=1) // white women, black women
3]     * ( black men - white men )      - (black women - white women)
        lincom (_b[2._at] - _b[1._at]) - (_b[4._at] - _b[3._at])
4]     est restore mymodel
5] }
```

Line 1 loops through outcome categories 1 through 5, assigning the value to the local `iout`. Line 2 makes four predictions for outcome ``iout`` and posts the estimates so they can be used by `lincom`. Line 3 computes the second differences by using `lincom`, and line 4 restores the `mlogit` estimation results. The output is 300 lines long. It could be shortened by adding the `noatlegend` option along with `mlistat` as shown above. Alternatively, we could use `quietly` to suppress the output.

As an alternative, we use `mllincom` instead of `lincom`, which allows us to create a compact table of results.


```

1) mlincom, clear
2) forvalues iout = 1/5 {
3)     quietly {
4)         margins, predict(outcome(`iout`)) post atmeans ///
           at(black=0 female=0) at(black=1 female=0) // white men, black men ///
           at(black=0 female=1) at(black=1 female=1) // white women, black women
5)         mlincom (2-1)-(4-3), save label(Outcome `iout`)
6)         est restore mymodel
7)     }
8) }

```

Line 1 clears the matrix where `mlincom` will accumulate results. Without this line, new results would be attached to results that might have been saved by `mlincom` run earlier. Lines 3 through 7 use `quietly` to suppress the display of results from `margins` and `mlincom`. Line 5 replaces `lincom` with `mlincom`, which lets us refer to the predictions by their position in the output of `margins`, rather than requiring the `_b[]` syntax. The `save` option adds the current results to the matrix holding prior results. `label()` adds labels to each set of results, in this case, indicating the outcome being tested. We run `mlincom` to list the results:

. mlincom				
	lincom	pvalue	ll	ul
Outcome 1	0.020	0.366	-0.023	0.063
Outcome 2	0.001	0.967	-0.037	0.039
Outcome 3	0.013	0.238	-0.008	0.034
Outcome 4	-0.004	0.836	-0.046	0.037
Outcome 5	-0.029	0.082	-0.061	0.004

The second differences are all less than 0.03 in magnitude and none are statistically significant. We conclude the following:

For those that are average on all characteristics, the marginal effect of race on party affiliation are the same for men and women.

8.9.2 (Advanced) Predictions using local means and subsamples

Comparing groups by making predictions with local means or by average predictions within subsamples is important for nuanced interpretations of group differences. To do this requires Stata programming and using the `over()` option with `margins`. Although you might want to skip this section on first reading, we encourage you to return to this section when you are comfortable with the commands used in other sections on interpretation with predictions.

To compute the predicted probabilities of party affiliation by race and gender, we used the `atmeans` option to hold all other variables at the means for the estimation sample. Accordingly, the predictions are comparing white men, black men, white women, and black women who have the same values for age, income, and education. Because the four race-gender groups are likely to differ on these variables, the predictions must be viewed as a “what if” experiment: what would happen if these groups had the same distributions of other characteristics?

We could compute predictions by using within-group means, which we refer to as local means. Using methods discussed on page 273, we use `if` conditions to select the sample for each `mtable` command:

```
. mlogit party age income i.black i.female i.educ, base(5)
(output omitted)
. estimates store mymodel
. qui mtable if black==0 & female==0, atmeans noci rowname(White Men) clear
. qui mtable if black==1 & female==0, atmeans noci rowname(Black Men) below
. qui mtable if black==0 & female==1, atmeans noci rowname(White Women) below
. mtable if black==1 & female==1, atmeans noci rowname(Black Women) below
Expression: Pr(party), predict(outcome())
```

	StrDem	Dem	Indep	Rep	StrRep
White Men	0.128	0.283	0.109	0.325	0.156
Black Men	0.461	0.310	0.168	0.044	0.017
White Women	0.145	0.354	0.093	0.298	0.109
Black Women	0.460	0.364	0.129	0.037	0.011

Specified values of covariates

	age	income	black	female	2. educ	3. educ
Set 1	45.1	43.5	0	0	.546	.324
Set 2	45	29.8	1	0	.518	.188
Set 3	47.1	35.2	0	1	.624	.222
Current	45.4	20.4	1	1	.59	.143

The values of the covariates show substantial differences among the groups, especially with respect to income, where white men have more than twice the average income of black women. Consequently, the predicted probabilities with local means differ from those computed with global means. For example, for black women, the probability of being a strong Democrat is 0.417 when global means are used compared with 0.460 when local means are used.

Although we can create the predictions we want by using `mtable` with `if` conditions, this approach does not allow us to compute first and second differences across the groups. Technically, the problem is that at the end of each `mtable` command (or `margins`, if we had used that command instead), only predictions for the current group can be posted to `e(b)` and `e(V)` for use by `lincom` or `margins`. A relatively efficient way to deal with this limitation is to use the `over(over-variables)` option. With the `over()` option, `margins` computes predictions based on the subsample of cases defined by the *over-variables*.

The *over-variables* can be any categorical variables in the dataset, even if they are not used in the regression model. For our purposes, we want to use `over(female black)` to compute predictions based on subsamples defined by race and gender:

```
. margins, over(female black) atmeans post
Adjusted predictions                                     Number of obs =      1382
Model VCE      : OIM
Expression     : Pr(party==StrDem), predict()
over          : female black
at            : 0.female#0.black

               age      =    45.13171 (mean)
               income    =    43.54512 (mean)
               black     =         0
               female    =         0
               1.educ    =    .1300813 (mean)
               2.educ    =    .5463415 (mean)
               3.educ    =    .3235772 (mean)

(output omitted)

               1.female#1.black
               age      =    45.3619 (mean)
               income    =    20.42619 (mean)
               black     =         1
               female    =         1
               1.educ    =    .2666667 (mean)
               2.educ    =    .5904762 (mean)
               3.educ    =    .1428571 (mean)
```

	Delta-method		z	P> z	[95% Conf. Interval]	
	Margin	Std. Err.				
female#black						
no#no	.1278946	.013109	9.76	0.000	.1022014	.1535878
no#yes	.4607531	.0430804	10.70	0.000	.376317	.5451892
yes#no	.1454926	.0143397	10.15	0.000	.1173872	.1735979
yes#yes	.4596925	.0411218	11.18	0.000	.3790952	.5402899

```
. estimates restore mymodel
(results mymodel are active now)
```

This provides the same estimates as the commands

```
margins if black==0 & female==0, atmeans post
estimates restore mymodel
margins if black==0 & female==1, atmeans post
estimates restore mymodel
margins if black==1 & female==0, atmeans post
estimates restore mymodel
margins if black==1 & female==1, atmeans post
estimates restore mymodel
```

except that `margins, over()` post posts the four predictions that allow us to compute and test second differences.

Now, we can use a `forvalues` loop to test whether second differences are equal to 0 for each outcome:


```

. mlincom, clear
. forvalues iout = 1/5 {
2.     quietly {
3.         margins, over(female black) predict(outcome(`iout`)) post atmeans
4.         mlincom (2-1)-(4-3), save label(Outcome `iout`)
5.         est restore mymodel
6.     } // end of quietly
7. } // end of forvalues

```

```

. mlincom

```

	lincom	pvalue	ll	ul
Outcome 1	0.019	0.415	-0.026	0.064
Outcome 2	0.019	0.345	-0.020	0.057
Outcome 3	0.024	0.047	0.000	0.048
Outcome 4	-0.020	0.360	-0.064	0.023
Outcome 5	-0.041	0.019	-0.075	-0.007

Gender differences in the effect of race are larger when computed using group-specific means for the other variables. The effect of race on being Independent is significantly larger for men than women, and the effect of race on being strongly Republican is significantly larger for women than men.

A related approach for comparing groups is to compute the average predicted probabilities within each of the subsamples defined by the groups being compared. For example, we can compare the average probability of being Republican for white men with the average probability for black men. To make these computations only requires us to remove the option `atmeans` from the commands used above.

```

. mlincom, clear
. forvalues iout = 1/5 {
2.     quietly {
3.         margins, over(female black) predict(outcome(`iout`)) post
4.         mlincom (2-1)-(4-3), save label(Outcome `iout`)
5.         est restore mymodel
6.     } // end of quietly
7. } // end of forvalues

```

```

. mlincom

```

	lincom	pvalue	ll	ul
Outcome 1	0.021	0.294	-0.018	0.060
Outcome 2	0.016	0.357	-0.018	0.051
Outcome 3	0.019	0.080	-0.002	0.041
Outcome 4	-0.015	0.476	-0.057	0.026
Outcome 5	-0.041	0.017	-0.075	-0.007

The results in this example lead to the same substantive conclusions obtained using local means.

8.10 Graphing predicted probabilities

Graphing predicted probabilities for each outcome can also be useful for the MNLM and is done exactly as it was for the ORM. To illustrate this, we create plots to show the effects of age and income on party affiliation. After fitting the model, we compute predictions as income increases from \$0 to \$100,000, holding other variables at their means.

```
. mlogit party age income i.black i.female i.educ, base(5) vsquish
(output omitted)
. mgen, atmeans at(income=(0(10)100)) stub(mnlmI) replace
Predictions from: margins, atmeans at(income=(0(10)100)) predict(outcome())
```

Variable	Obs	Unique	Mean	Min	Max	Label
mnlmIpr1	11	11	.1604032	.090648	.2445128	pr(y=StrDem) from margins
mnlmIpr2	11	11	.1253144	.0479288	.1908274	95% lower limit
mnlmIpr3	11	11	.1954919	.1333672	.2981981	95% upper limit
mnlmIincome	11	11	50	0	100	Income in \$1,000s

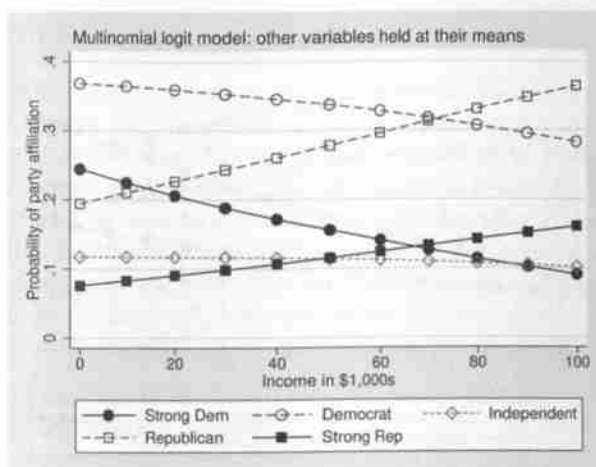
```
(output omitted)
```

```
Specified values of covariates
```

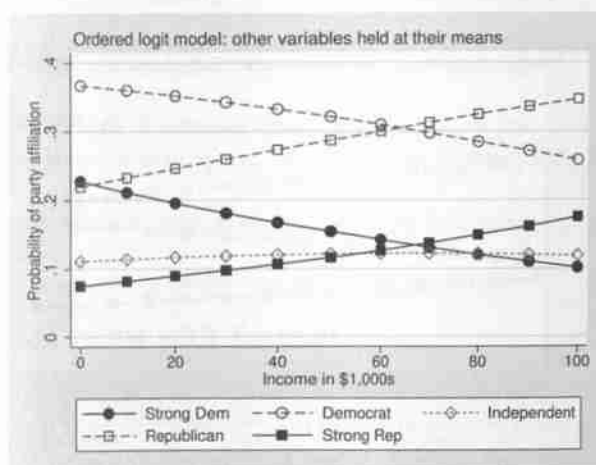
	1.	1.	2.	3.
age	black	female	educ	educ
45.94645	.1374819	.4934877	.5803184	.2590449

Using variable labels to assign labels to the lines, we can plot the predictions with these commands:

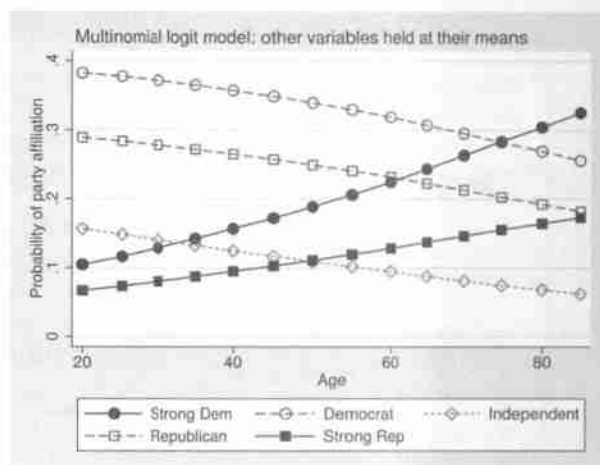
```
. label var mnlmIpr1 "Strong Dem"
. label var mnlmIpr2 "Democrat"
. label var mnlmIpr3 "Independent"
. label var mnlmIpr4 "Republican"
. label var mnlmIpr5 "Strong Rep"
. graph twoway connected
> mnlmIpr1 mnlmIpr2 mnlmIpr3 mnlmIpr4 mnlmIpr5 mnlmIincome,
> title("Multinomial logit model: other variables held at their means",
> pos(11) size(medium))
> ytitle(Probability of party affiliation) ylab(0(.1).4, grid gmax gmin)
> msym(0 Oh dh sh s) mcol(gs1 gs5 gs8 gs5 gs1)
> lpat(solid dash shortdash dash solid) lcol(gs1 gs5 gs8 gs5 gs1)
> legend(rows(2))
```

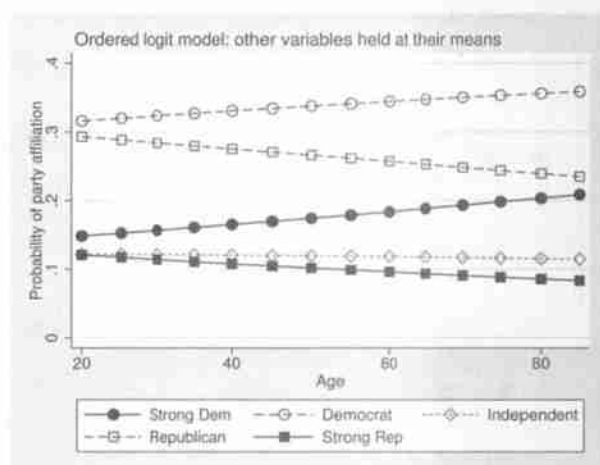
The probabilities of being a strong Democrat (●) or a Democrat (○) decrease with income, while the probabilities of being a Republican (□) or a strong Republican (■) increase, with little change in the probability of being Independent (◇). Using `ologit` to fit the ordinal model, we obtain a nearly identical graph:



The results are quite different when we examine the effect of age on party affiliation. For the MNLM, we plot predictions as age increases from 20 to 85, holding other variables at their means (commands not shown):



The probabilities for the two strong affiliations, shown with solid markers, both increase with age. This pattern could not be obtained with an ORM that requires that changes in the extreme categories be in opposite directions. To see this, consider the corresponding graph based on predictions from the OLM:



This example illustrates the risk of assuming an ORM is appropriate simply because the dependent variables can be ordered. Although income affects affiliation as would be expected with a unidimensional, ordinal outcome, age increases the strength of affiliation but does not affect left-right orientation. When using an ordinal model, we believe it is good practice to examine the sensitivity of the results to the constraints of ordinality by comparing the results from the ordinal model with those from the MNLM or the generalized OLM.

8.11 Odds ratios

Discrete change coefficients do not show the dynamics among the outcomes. For example, being black increases the probability of being a Democrat or an Independent, but how does it affect the probability of being Democrat relative to being Independent? To deal with these questions, odds ratios, also referred to as relative-risk ratios and factor change coefficients, can be used to explore how variables affect the choice of one outcome compared with another outcome. Odds ratios do not provide a complete picture of the effects of variables on the outcomes and have the same limitations discussed for the binary logit model in chapter 6; however, in the MNLM, odds ratios complement the information provided by marginal effects and other types of predictions.

The factor change in the odds of outcome m versus outcome n as x_k increases by δ , holding other variables constant, equals

$$\frac{\Omega_{m|n}(\mathbf{x}, x_k + \delta)}{\Omega_{m|n}(\mathbf{x}, x_k)} = e^{\beta_{k,m|n}\delta}$$

If the amount of change is $\delta = 1$, the odds ratio can be interpreted as follows:

For a unit increase in x_k , the odds of m versus n are expected to change by a factor of $\exp(\beta_{k,m|n})$, holding all other variables constant.

If the amount of change is $\delta = s_{x_k}$, then the odds ratio can be interpreted as follows:

For a standard deviation increase in x_k , the odds of m versus n are expected to change by a factor of $\exp(\beta_{k,m|n} \times s_k)$, holding all other variables constant.

Other values of δ can also be used, such as $\delta = 4$ for four years of education or $\delta = 10$ for \$10,000 in income.

8.11.1 Listing odds ratios with listcoef

The difficulty in interpreting odds ratios for the MNLM is that to understand the effect of a variable, you need to examine the coefficients for comparisons among all pairs of outcomes. The standard output from `mlogit` includes only a minimal set of $J - 1$ comparisons with the base outcome. Although you could estimate coefficients for all possible comparisons by rerunning `mlogit` with different bases (for example, `mlogit party female black..., base(1)`; `mlogit party female black..., base(2)`; etc.), using `listcoef` is simpler. For example, to examine the odds ratios for variable `black`, type


```
. use partyid4, clear
(partyyid4.dta | 1992 American National Election Study | 2014-03-12)
. mlogit party age income i.black i.female i.educ, base(5)
(output omitted)
. listcoef black, help
mlogit (N=1382): Factor change in the odds of party
Variable: 1.black (sd=0.344)
```

		b	z	P> z	e ^b	e ^b StdX
StrDem	vs Dem	0.9963	5.022	0.000	2.708	1.409
StrDem	vs Indep	0.7845	3.052	0.002	2.191	1.310
StrDem	vs Rep	2.9692	7.666	0.000	19.475	2.781
StrDem	vs StrRep	3.0754	5.091	0.000	21.659	2.885
Dem	vs StrDem	-0.9963	-5.022	0.000	0.369	0.709
Dem	vs Indep	-0.2118	-0.830	0.407	0.809	0.930
Dem	vs Rep	1.9728	5.132	0.000	7.191	1.973
Dem	vs StrRep	2.0791	3.448	0.001	7.997	2.047
Indep	vs StrDem	-0.7845	-3.052	0.002	0.456	0.763
Indep	vs Dem	0.2118	0.830	0.407	1.236	1.076
Indep	vs Rep	2.1846	5.220	0.000	8.887	2.122
Indep	vs StrRep	2.2909	3.658	0.000	9.884	2.202
Rep	vs StrDem	-2.9692	-7.666	0.000	0.051	0.360
Rep	vs Dem	-1.9728	-5.132	0.000	0.139	0.507
Rep	vs Indep	-2.1846	-5.220	0.000	0.113	0.471
Rep	vs StrRep	-0.1063	-0.155	0.877	1.112	1.037
StrRep	vs StrDem	-3.0754	-5.091	0.000	0.046	0.347
StrRep	vs Dem	-2.0791	-3.448	0.001	0.125	0.489
StrRep	vs Indep	-2.2909	-3.658	0.000	0.101	0.454
StrRep	vs Rep	-0.1063	-0.155	0.877	0.899	0.964

```
b = raw coefficient
z = z-score for test of b=0
P>|z| = p-value for z-test
eb = exp(b) = factor change in odds for unit increase in X
ebStdX = exp(b*SD of X) = change in odds for SD increase in X
```

The odds ratios of interest are in the column labeled e^b. For example, the odds ratio for black for the outcomes StrDem vs Dem is 2.708, which is significant at the 0.001 level. It can be interpreted as follows:

Being black increases the odds of having a strong Democratic affiliation compared with a Democratic affiliation by a factor of 2.7, holding other variables constant.

Even for a single variable, there are a lot of coefficients; we often hear people lament that there are “too many” coefficients to interpret. Fortunately, a simple graph makes this task manageable, even for complex models.

8.11.2 Plotting odds ratios

An odds-ratio plot lets you quickly see patterns in coefficients, even for complex models with many outcomes. Methods for plotting odds ratios were developed by Long (1987)

while using the MNLM to examine factors that determine the organizational contexts in which scientists work (Long and McGinnis 1981). Although an odds-ratio plot was included in that paper, this is generally not the most effective way to use these graphs. Rather, the graphs provide a quick way to assess all the parameters in the model and to get a general sense of what is going on to help plan further analyses. Experience in teaching suggests that within an hour, students gain a “feel” for these graphs that allows them to evaluate the results of an MNLM in only a few minutes. Building on these insights, more detailed analyses can be planned using other methods of interpretation.

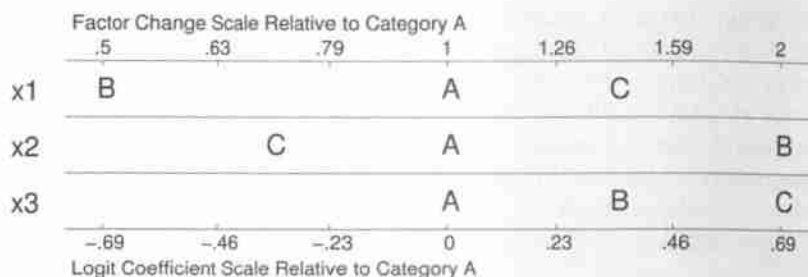
To explain how to interpret an odds-ratio plot, we begin with hypothetical results from an MNLM with three outcomes and three independent variables:

Comparison		Logit coefficients		
		x_1	x_2	x_3
$B \mid A$	$\beta_{B A}$	-0.693	0.693	0.347
	$\exp(\beta_{B A})$	0.500	2.000	1.414
	p	0.04	0.01	0.42
$C \mid A$	$\beta_{C A}$	0.347	-0.347	0.693
	$\exp(\beta_{C A})$	1.414	0.707	2.000
	p	0.21	0.04	0.37
$C \mid B$	$\beta_{C B}$	1.040	-1.040	0.346
	$\exp(\beta_{C B})$	2.828	0.354	1.414
	p	0.02	0.03	0.21

These coefficients were constructed to have specific types of relationships among outcomes and variables:

- The β coefficients for x_1 and x_2 on $B \mid A$ (which you can read as B versus A) are equal but of opposite sign. The coefficient for x_3 is half as large.
- The β coefficients for x_1 and x_2 on $C \mid A$ are half as large and in opposite directions as the coefficients on $B \mid A$, whereas the coefficient for x_3 is in the same direction but is twice as large.

In an odds-ratio plot, each independent variable is presented on a separate row, with the horizontal axis indicating the magnitude of the β coefficients associated with each contrast of outcomes. Here is the plot, where the letters correspond to the outcome categories:



We now explain how the graph conveys the information from the table of coefficients.

Sign of coefficients. If a letter is to the right of another letter, increases in the independent variable make the outcome to the right more likely relative to outcomes located to the left, holding all other variables constant. Thus relative to outcome A , an increase in x_1 increases the odds of C and decreases the odds of B . This corresponds to the positive sign of $\beta_{1,C|A}$ and the negative sign of $\beta_{1,B|A}$. The signs of these coefficients are reversed for x_2 , and accordingly, the plot for x_2 is a mirror image of that for x_1 .

Magnitude of coefficients. The distance between a pair of letters indicates the magnitude of the coefficient. The additive scale on the bottom axis measures the value of the $\beta_{k,m|n}$'s. The multiplicative scale on the top axis measures the odds ratios $\exp(\beta_{k,m|n})$. For both x_1 and x_2 , the distance between A and B is twice the distance between A and C , which reflects that $\beta_{1,B|A}$ is twice as large as $\beta_{1,C|A}$ and $\beta_{2,B|A}$ is twice as large as $\beta_{2,C|A}$. For x_3 , the distance between A and B is half the distance between A and C , reflecting that $\beta_{3,C|A}$ is twice as large as $\beta_{3,B|A}$.

The additive relationship. The additive relationships among coefficients in (8.1) are shown in the graph. For all the independent variables, $\beta_{k,C|A} = \beta_{k,B|A} + \beta_{k,C|B}$. Accordingly, the distance from letters A to C in the graph is the sum of the distances from A to B and B to C . This is easiest to see in the row for variable x_3 , where all the coefficients are positive. By plotting the $J - 1$ coefficients from a minimal set, it is possible to visualize the relationships among all pairs of outcomes.

The base outcome. In the graph above, the A s are aligned vertically because the plot uses A as the base outcome when graphing the coefficients. The choice of the base is arbitrary. We could have used alternative B as the base instead, which would shift the rows of the graph to the left or right so that the B s lined up. Doing this leads to the following graph:

Factor Change Scale Relative to Category B							
	.35	.5	.71	1	1.41	2	2.83
x1				B		A	C
x2	C	A		B			
x3			A	B	C		
	-1.04	-.69	-.35	0	.35	.69	1.04
Logit Coefficient Scale Relative to Category B							

Creating odds-ratio plots with `mlogitplot`

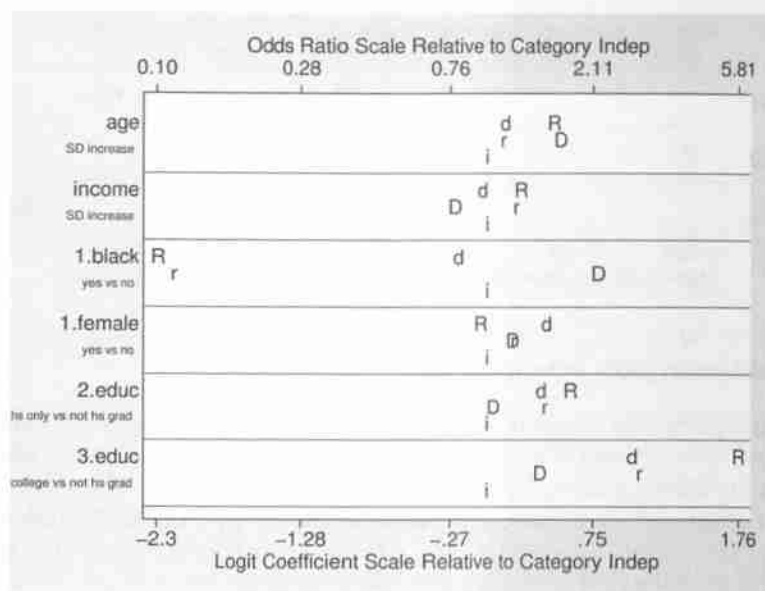
Odds-ratio plots can be easily constructed with the `SPost` command `mlogitplot`.⁴ In this section, we construct a series of odds-ratio plots that illustrate how this command can be used to understand the factors affecting party affiliation. We assume that the results from `mlogit` are in memory. Because `mlogitplot` does not change the estimation results, there is no need to use `estimates store` and `estimates restore`. Full details on the syntax for `mlogitplot` are given in `help mlogitplot`.

As a first step in examining results from an MNLM, it is useful to create an odds-ratio plot for all variables. This is done by typing the command `mlogitplot`. To make our graph more effective, we add a few options:

```
mlogitplot, symbols(D d i r R) base(3) linepvalue(1) leftmargin(2)
```

The option `symbols(D d i r R)` labels the outcomes, and `base(3)` specifies to line up the outcomes on base 3, which is `Indep` shown by `i`. The option `linepvalue(1)` removes lines indicating statistical significance, an important feature that will be discussed soon. Finally, `leftmargin(2)` adds space on the left for the labels associated with factor variables; in practice, you will need to experiment to determine how large this margin needs to be for your plot. The following graph is created:

4. The command `mlogitplot` in `SPost13` replaces the commands `mlogview` and `mlogplot` in `SPost9`.



The independent variables are listed on the left, with the vertical distance for a given variable used simply to prevent the symbols from overlapping. The default vertical distance does not always work (for example, notice how “r” is inside the “D” for variable *1.female*), and later we consider options for refining these offsets.

From the plot, we immediately see that the odds ratios for **black** are the largest, increasing the odds of being a strong Democrat (“D”), a Democrat (“d”), or an Independent (“i”) compared with being Republican (“r”) or strong Republican (“R”). Having a college education compared with not completing high school (*3.educ*) increases the odds of being a strong Republican (“R”) relative to the other categories. Consistent with our findings when plotting probabilities against age and income, age increases the odds of strongly affiliating with either party relative to affiliations that are less strong, while income increases the odds of more right-leaning affiliations relative to left-leaning.

The current graph has two limitations. First, while it shows the size of odds ratios, it does not indicate whether they are statistically significant. While it is tempting to assume that larger odds ratios imply smaller *p*-values for testing the hypothesis that the odds ratio is 1, that should not be done! Instead, we need to add the significance level to the graph. Second, because a large odds ratio does not necessarily correspond to a large marginal effect, we will add information on marginal effects to the graph.

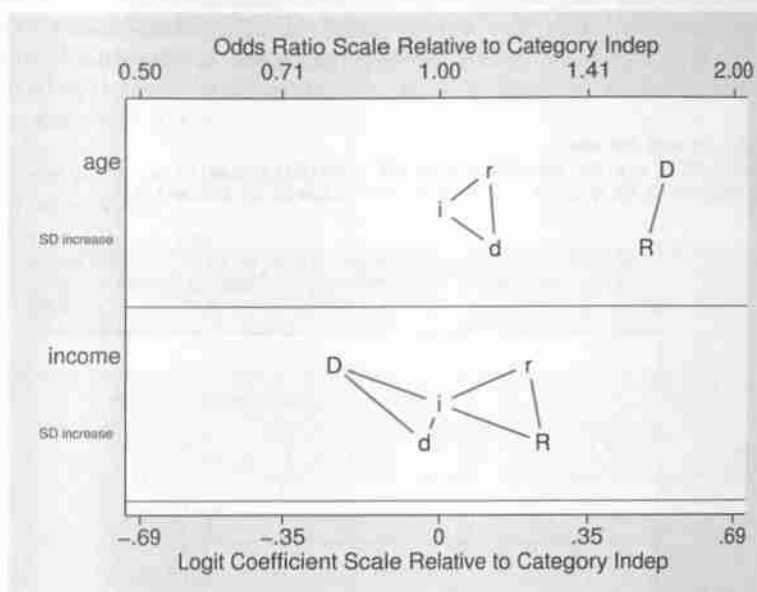
Adding significance levels

The distance between two outcomes indicates the magnitude of the coefficient. If a coefficient is not significantly different from 0, we add a line connecting the two outcomes, suggesting that those outcomes are “tied together”. By default, `mlogitplot`

connects outcomes where the odds ratio has a p -value greater than 0.10. You can choose other p -values with the `linepvalue(#)` option, and you can remove all lines by specifying `linepvalue(1)`.

To illustrate how statistical significance is added to the graph, we plot the odds ratios for age and income:

```
. mlogitplot age income,
> symbols(D d i r R) base(3) ormin(.5) ormax(2) nticks(5)
> offsetlist(2 -2 0 2 -2 2 -2 0 2 -2) ysize(2.4) scale(1.1)
```



Based on this graph, we conclude the following:

Age significantly increases the odds of affiliations that are strong relative to those that are not strong, with no significant odds ratios differentiating outcomes within these two groups.

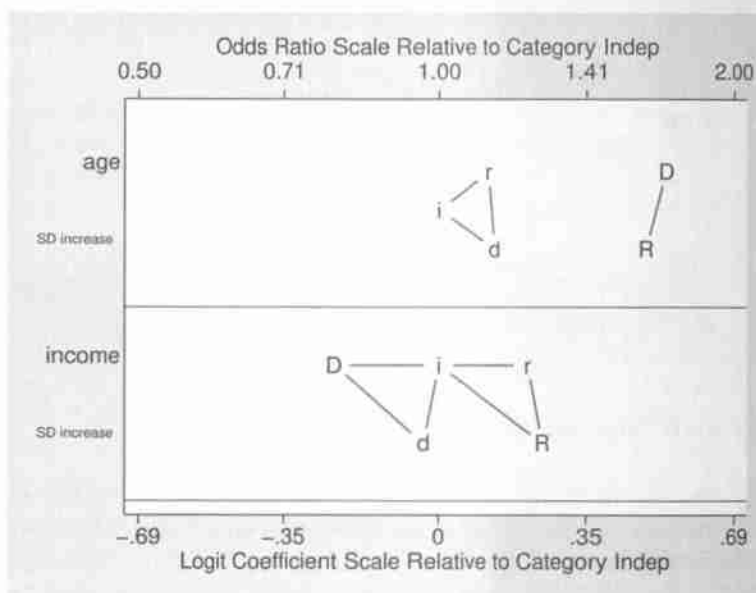
For income, we conclude the following:

In terms of the conventional left–right continuum, income increases the odds of affiliations to the right but does not significantly differentiate affiliations that are adjacent, such as strong Democrat compared with Democrat or Republican compared with strong Republican.

Sometimes the lines do not clearly show whether an odds ratio is significant. For example, in the last plot, it is not clear whether there is a line connecting “i” and “d”

for income because the symbols are so close to one another. There are several ways to resolve this. First, we can examine the odds ratio in the output from `listcoef`, where we see that the effect is not significant. Second, we can reduce the size of the spacing between symbols and the start of the lines by using the `linegapfactor(#)` option. If we add `linegapfactor(.5)`, we would see that there is a line connecting “i” and “d”. Finally, we can revise the vertical offsets used for placing letters with the `offsetlist()` option. Offsets are determined by specifying one integer in the range from -5 to 5 for each outcome for each variable in the graph. By default, the offsets are 2 -2 0 2 -2 0 2 -2 0 To adjust our prior graph for income, we use 2 -2 1 2 -2 to move “i” up because its offset has been increased from 0 to 1. Remember that these adjustments have no substantive meaning; they simply make the information clearer. Using these offsets, the following command creates a graph that makes it clear that “i” and “D” are linked:

```
. mlogitplot age income,
> symbols(D d i r R) base(3) ormin(.5) ormax(2) nticks(5)
> offsetlist(2 -2 0 2 -2 2 -2 2 2 -2) ysize(2.4) scale(1.1)
```

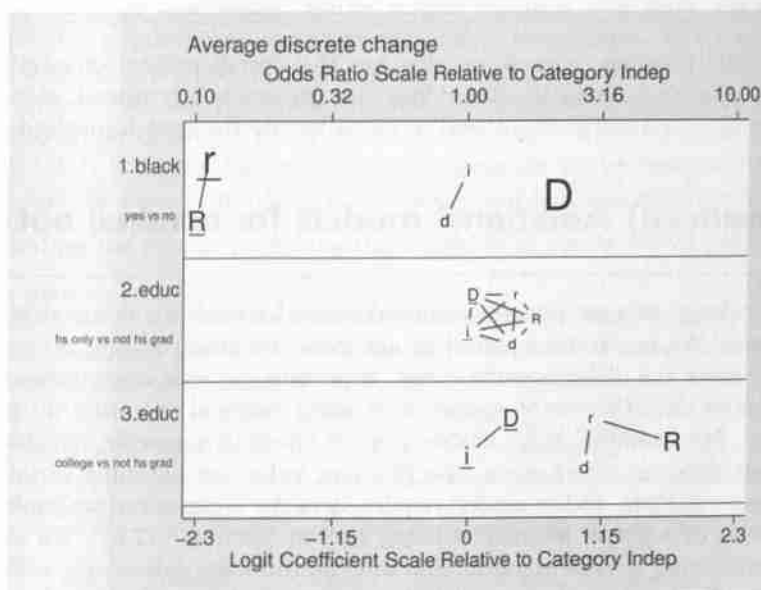


The plot for income illustrates why it is important to examine all contrasts (that is, odds ratios), not just the minimal set. Suppose that we had fit the model with base outcome 3, corresponding to “i” in the graph. At the 0.10 level, none of the odds ratios relative to “i” are significant, as indicated by the lines from “i” to each of the other outcomes. It would be incorrect to assume that income did not significantly affect party affiliation because the contrasts for outcomes “D” versus “r”; “D” versus “R”; “d” versus “r”; and “d” versus “R” are significant. An odds-ratio plot is a quick way to see all the contrasts.

Adding discrete change

In chapter 6, we emphasized that whereas the factor change in the odds is constant across the levels of all variables, the marginal effect gets larger or smaller at different values of the variables. For example, if the odds increase by a factor of 10 but the current odds are 1 in 10,000, the change in the probability is small. But if the current odds are 1 in 5, the change in probability is large. Information on the change in probability can be incorporated into the odds-ratio plot by making the area of a square drawn around a letter proportional to the discrete change in the probability. Sign is indicated by underlining the letter if the marginal effect is negative. To add this information, we must first run `mchange` to calculate marginal effects for the amount of change we are interested in (for example, 1 unit or a standard deviation). Second, we add the option `mchange` to `mlogitplot`. To illustrate this, we plot the odds ratios and average discrete changes for `educ` and `black`:

```
. mchange black educ, amount(sd)
(output omitted)
. mlogitplot black educ,
> symbols(D d i r R) base(3) ormin(.1) ormax(10) ntics(5)
> mchange subtitle(Average discrete change, position(11))
> offsetlist(0 -2 2 3 -2 2 -2 -1 2 0 2 -2 -1 2 0) leftmargin(12)
```



By far, the largest marginal effect is the increase in the probability of being a strong Democrat ("D") if you are black. In terms of the odds ratios, race divides affiliations into three groups: 1) strong Republicans ("R") and Republicans ("r"); 2) Democrats ("d") and Independents ("i"); and 3) strong Democrats ("D").

By examining both the odds ratio and the discrete change, it is clear that a positive odds ratio for outcome *A* compared with outcome *B* does not indicate the sign of the discrete changes for the two outcomes. For example, with an odds ratio greater than 1, the discrete changes for both outcomes can both be positive, can both be negative, or can differ in sign. An odds ratio indicates the ratio of the change of one category relative to another, not the direction of the change. The graph also illustrates what is found when a variable has no significant effects, as is the case for `2.educ`. The discrete changes are small and all letters are connected by lines, indicating that none of the odds ratios are significant.

The size of symbols. When the `mchange` option is used, the size of the symbol reflects the size of the effect. Because many letters are more or less square, the size of the area of a square drawn around the symbol is proportional to the absolute magnitude of the marginal effect. This can be misleading in some cases. For example, the letters “r” and “R” both represent the same size effect, but “R” is larger. As long as you keep this in mind, the size of the letters should give you a rough idea of the magnitude of effects. If you want to be certain, check the output from `mchange`.

With a little practice, you can quickly see the overall pattern of relationships in your model by using odds-ratio plots. Once the pattern is determined, other methods of interpretation can be effectively used to demonstrate the most important findings.

8.12 (Advanced) Additional models for nominal outcomes

This (long) section presents some additional models for nominal outcomes. We mark the material as advanced because you may wish to only skim the different subsections, especially because several require types of data that most applications using nominal outcomes do not have. For example, some models require alternative-specific variables, where different alternatives have different values on the same variable (section 8.12.4). Other models require that the alternatives are ranked instead of a single alternative being chosen (section 8.12.5). We also provide some details about models that we think are didactically useful for understanding the overall logic of how modeling categorical outcomes is done, such as showing how the conditional logit model can be used to produce the same results as the MNLM (8.12.2), but we do not advocate fitting the model this way because the MNLM is simpler.

8.12.1 Stereotype logistic regression

In the last chapter, we postponed a detailed discussion of the stereotype logistic regression model (SLM) even though the model can be considered a model for ordinal outcomes. We did this because the SLM is easier to understand once you are familiar with the MNLM. The SLM, proposed by Anderson (1984), is more flexible than the OLM because it does not require the proportional-odds assumption, yet it can be more parsimonious than the MNLM. Anderson developed the SLM in reaction to the limitations of the OLM, which he referred to as the grouped continuous regression model. The term “grouped continuous” reflected that the OLM can be motivated by a continuous, latent variable that is divided (that is, grouped) by thresholds that lead to the observed categories. In contrast, Anderson thought of the outcome categories in the SLM as “assessed”. Each respondent is considered to have stereotypes that characterize the outcome choices. The respondent assesses each outcome and then picks the alternative whose stereotype most closely matches the respondent’s views on the question being asked. Although this explains the name of the model, there is no reason to limit the application of the model to outcomes that are generated in an assessed fashion.

Although the SLM is more parsimonious in the number of parameters than the MNLM, we will show that the full interpretation of the SLM is as complicated as that of the MNLM. If the full complexity of the model is not considered, valuable information can be lost and incorrect conclusions can be made. There is also some confusion about whether the SLM requires the dependent variable to be ordered. In its simplest form, the SLM orders the dependent variable along one dimension, but the outcomes are not necessarily ordered the way you think they are. For example, you might think that your outcomes should be ordered 1 2 3 4, but the SLM might determine that the ordering should be 2 1 3 4. In higher-dimensional SLMs, categories are ordered on more than one dimension, and the idea of ordinality is lost as the SLM becomes identical to the MNLM.

We introduce the SLM by reviewing the results from our MNLM of party affiliation:

```
. use partyid4, clear
(partiid4.dta | 1992 American National Election Study | 2014-03-12)
. mlogit party age income i.black i.female i.educ, base(5) nolog vsquish
```

Multinomial logistic regression

Number of obs	=	1382
LR chi2(24)	=	311.25
Prob > chi2	=	0.0000
Pseudo R2	=	0.0735

Log likelihood = -1960.9107

party	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
StrDem						
age	.0028185	.00644	0.44	0.662	-.0098036	.0154407
income	-.0174695	.0045777	-3.82	0.000	-.0264416	-.0084974

(output omitted)

Dem						
age	-.0207981	.0059291	-3.51	0.000	-.032419	-.0091772
income	-.0101908	.0035532	-2.87	0.004	-.0171549	-.0032267

(output omitted)

<hr/>						
Indep						
age	-.0287992	.0074315	-3.88	0.000	-.0433648	-.0142337
income	-.0089716	.0047821	-1.88	0.061	-.0183443	.0004012
(output omitted)						
<hr/>						
Rep						
age	-.0217144	.0060422	-3.59	0.000	-.0335569	-.0098718
income	-.0012715	.0033629	-0.38	0.705	-.0078627	.0053196
(output omitted)						
<hr/>						
StrRep	(base outcome)					
<hr/>						

```
. predict mn1m_1 mn1m_2 mn1m_3 mn1m_4 mn1m_5
(option pr assumed; predicted probabilities)
```

If *party* is ordinal with respect to the independent variables, what pattern would we expect for the β 's in the four equations? The top panel, labeled *StrDem*, presents coefficients for the equation comparing outcome *StrDem* with the base outcome *StrRep*; the panel *Dem* presents coefficients comparing *Dem* with *StrRep*; and so on. If an increase in an explanatory variable increases the odds of answering *StrDem* versus *StrRep*, we would also expect the variable to increase the odds of *Dem* versus *StrRep*, as well as increase the odds of *Indep* and *Rep* versus *StrRep*. That is, we would expect $\beta_{k,SD|SR}$, $\beta_{k,D|SR}$, $\beta_{k,I|SR}$, and $\beta_{k,R|SR}$ to have the same sign. More than this, we would expect the coefficient to be largest when comparing categories *StrDem* and *StrRep*, which are furthest apart on the ordinal ranking from *StrDem* to *StrRep*, and smallest for adjacent categories, such as *Rep* and *StrRep*. In the *mlogit* output above, this means that if *party* is ordinal, we would expect the coefficients in the panel labeled *StrDem* to be the largest (either positive or negative), followed by those in the *Dem* panel, with the smallest found in the *Rep* panel. Although this pattern holds for *income*, the estimates for *age* violate the pattern.

A further possibility is that the magnitudes of $\beta_{k,SD|SR}$, $\beta_{k,D|SR}$, $\beta_{k,I|SR}$, and $\beta_{k,R|SR}$ are not just consistently ordered for each independent variable but the relative magnitudes of these coefficients are the same for all independent variables. The implication is that the distance or difficulty in moving from *StrDem* to *Dem* compared with moving from *Rep* to *StrRep* is the same for each variable. For example, if $\beta_{age,SD|D}$ is 2.7 times larger than $\beta_{age,R|SR}$, then $\beta_{income,SD|D}$ would be 2.7 times larger than $\beta_{income,D|SA}$, and so on. This implies that there is a coefficient $\tilde{\beta}_k$ for each x_k and scaling parameters ϕ_j for each outcome j such that for each x_k

$$\beta_{k,SD|SR} = \phi_{SD} \tilde{\beta}_k$$

$$\beta_{k,D|SR} = \phi_D \tilde{\beta}_k$$

$$\beta_{k,I|SR} = \phi_I \tilde{\beta}_k$$

$$\beta_{k,R|SA} = \phi_R \tilde{\beta}_k$$

If these constraints are applied to the MNLM, you have the one-dimensional SLM. That is, the SLM is an MNLM fit with constraints. If the constraints adequately characterize the data-generating process, the more parsimonious SLM should fit the data nearly as well as the MNLM.

Formal statement of the one-dimensional SLM

With these ideas in mind, we can present the model more formally. To simplify the presentation, we assume that there are three outcomes and two independent variables. For the MNLM with base outcome 3,

$$\Pr(y = m | \mathbf{x}) = \frac{\exp(\beta_{0,m|3} + \beta_{1,m|3}x_1 + \beta_{2,m|3}x_2)}{\sum_{j=1}^3 \exp(\beta_{0,j|3} + \beta_{1,j|3}x_1 + \beta_{2,j|3}x_2)} \text{ for } m = 1, 2 \quad (8.3)$$

For the SLM, using similar notation,

$$\Pr(y = m | \mathbf{x}) = \frac{\exp(\phi_m \tilde{\beta}_0 x_0 + \phi_m \tilde{\beta}_1 x_1 + \phi_m \tilde{\beta}_2 x_2)}{\sum_{j=1}^3 \exp(\phi_j \tilde{\beta}_0 x_0 + \phi_j \tilde{\beta}_1 x_1 + \phi_j \tilde{\beta}_2 x_2)} \text{ for } m = 1, 2 \quad (8.4)$$

For both models, $\Pr(y = 3 | \mathbf{x}) = 1 - \Pr(y = 1 | \mathbf{x}) - \Pr(y = 2 | \mathbf{x})$. The only difference between (8.3) and (8.4) is that $\beta_{k,m|3}$ is replaced by $\phi_m \tilde{\beta}_k$. This replacement forces the ratio of coefficients to be equal across variables. Specifically,

$$\frac{\phi_j \tilde{\beta}_1}{\phi_m \tilde{\beta}_1} = \frac{\phi_j \tilde{\beta}_2}{\phi_m \tilde{\beta}_2} = \frac{\phi_j}{\phi_m}$$

By comparison, in the MNLM, the ratio $\beta_{1,j|3}/\beta_{1,m|3}$ might be similar to $\beta_{2,j|3}/\beta_{2,m|3}$, but the model does not require this.

Because some of the parameters in the SLM are not identified, we must add constraints before the parameters can be estimated. To understand the identification constraints used by Stata, we find it helpful to compare them with the identifying constraints used for the MNLM. In the MNLM, we assume that $\beta_{k,3|3} = 0$, where 3 is the base outcome. This constraint simply says that a change in x_k does not change the odds of outcome 3 compared with outcome 3. The corresponding constraint in the SLM is $\phi_3 \tilde{\beta}_k = 0$. We assume that $\phi_3 = 0$ because we do not want to require $\tilde{\beta}_k = 0$, which would eliminate the effect of x_k for all pairs of outcomes. This is our first identification constraint. To understand the next constraint, we compare $\beta_{k,1|3}$ and $\beta_{k,2|3}$ for x_k in the MNLM with the corresponding pairs of coefficients $\phi_1 \tilde{\beta}_k$ and $\phi_2 \tilde{\beta}_k$ in the SLM. There are two free parameters $\beta_{k,1|3}$ and $\beta_{k,2|3}$ in the MNLM, but three parameters ϕ_1 , ϕ_2 , and $\tilde{\beta}_k$ in the SLM. To eliminate the "extra" parameter, we assume that $\phi_1 = 1$. With these constraints, the SLM is identified. These constraints are the defaults used by `slogit`, but `slogit` allows you to use other identifying constraints.

The notation we have used highlights the similarities between the MNLM and the SLM but differs from the notation used by Stata. To switch notations, we define $\theta_m \equiv \phi_m \tilde{\beta}_0$,

where $\theta_3 = 0$, and $\phi_m \beta_1 \equiv -\phi_m \tilde{\beta}_1$, where $\phi_3 = 0$ and $\phi_1 = 1$. Because the sign has changed, a positive coefficient in the MNLM corresponds to a negative coefficient in the SLM. With this new notation, we can write the model as

$$\Pr(y = m \mid \mathbf{x}) = \frac{\exp(\theta_m - \phi_m \beta_1 x_1 - \phi_m \beta_2 x_2)}{\sum_{j=1}^3 \exp(\theta_j - \phi_j \beta_1 x_1 - \phi_j \beta_2 x_2)} \quad (8.5)$$

We can generalize this equation to J outcomes and K independent variables,

$$\Pr(y = m \mid \mathbf{x}) = \frac{\exp(\theta_m - \phi_m \mathbf{x} \boldsymbol{\beta})}{\sum_{j=1}^J \exp(\theta_j - \phi_j \mathbf{x} \boldsymbol{\beta})} \quad (8.6)$$

where $\theta_J = 0$, $\phi_J = 0$, and $\phi_1 = 1$ are used to identify the model.

Fitting the SLM with slogit

The SLM is fit with the following command and its basic options:

```
slogit depvar [indepvars] [if] [in] [weight] [, dimension(#)
      baseoutcome(#) constraints(#) nocorner vce(vcetype) ]
```

Options

dimension(#) specifies the dimension of the model. The default is **dimension**(1). The maximum is either one less than the number of categories in the dependent variable or the number of explanatory variables, whichever is fewest. The dimension of an SLM is discussed below.

baseoutcome(#) specifies the outcome category whose associated θ and ϕ estimates will be constrained to 0. By default, this is the highest numbered category.

vce(*vcetype*) specifies the type of standard errors to be computed. See section 3.1.9 for details.

For other options, see [R] **slogit**.

Example of SLM

For our example of party affiliation, we fit the one-dimensional SLM and compute predictions:


```
. slogit party age income i.black i.female i.educ, vsquish
Iteration 0:  log likelihood = -2233.9118   (not concave)
Iteration 1:  log likelihood = -2033.644   (not concave)
Iteration 2:  log likelihood = -2008.6318   (not concave)
Iteration 3:  log likelihood = -2003.3438
Iteration 4:  log likelihood = -1997.4062
Iteration 5:  log likelihood = -1995.2213
Iteration 6:  log likelihood = -1995.0957
Iteration 7:  log likelihood = -1995.0948
Iteration 8:  log likelihood = -1995.0948

Stereotype logistic regression               Number of obs   =       1382
                                           Wald chi2(6)    =       103.46
Log likelihood = -1995.0948                 Prob > chi2     =       0.0000
```

```
( 1)  [phi1_1]_cons = 1
```

party	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
age	-.007771	.0059617	-1.30	0.192	-.0194556	.0039137
income	.0173648	.003745	4.64	0.000	.0100247	.0247049
black						
yes	-3.105455	.4465568	-6.95	0.000	-3.98069	-2.230219
female						
yes	-.1695415	.177839	-0.95	0.340	-.5180996	.1790166
educ						
hs only	.6146554	.2745719	2.24	0.025	.0765044	1.152806
college	1.276616	.3472982	3.68	0.000	.5959244	1.957308
/phi1_1	1	(constrained)				
/phi1_2	.626937	.0676574	9.27	0.000	.4943309	.7595431
/phi1_3	.7318878	.0751227	9.74	0.000	.5846499	.8791256
/phi1_4	.1636441	.0952039	1.72	0.086	-.0229521	.3502402
/phi1_5	0	(base outcome)				
/theta1	.9796152	.4466317	2.19	0.028	.1042332	1.854997
/theta2	1.458878	.3080763	4.74	0.000	.8550591	2.062696
/theta3	.4512317	.3519666	1.28	0.200	-.2386102	1.141074
/theta4	.9631935	.1777725	5.42	0.000	.6147659	1.311621
/theta5	0	(base outcome)				

```
(party=StrRep is the base outcome)
```

```
. predict slm1_1 slm1_2 slm1_3 slm1_4 slm1_5
```

```
(option pr assumed; predicted probabilities)
```

We show the iteration log to illustrate that the SLM often takes more steps to converge than the corresponding MNLM. The top panel contains estimates of the β 's. The next panel contains estimates of the ϕ 's, where the constraints are shown for $\phi_1 = 1$ and $\phi_5 = 0$. Notice that the ϕ 's are not ordered from largest to smallest. This means that the fit of the model is better if the outcomes are given a different ordering than that implied by the way party is numbered with 1 = StrDem, 2 = Dem, 3 = Indep, 4 = Rep, and 5 = StrRep. The intercepts θ are shown in the last panel, including the constraint $\theta_5 = 0$.

Interpretation using odds ratios

Because the SLM is a logit model, we can write the model as

$$\ln \Omega_{q|r}(\mathbf{x}) = \ln \frac{\Pr(y = q|\mathbf{x})}{\Pr(y = r|\mathbf{x})} = (\theta_q - \theta_r) - (\phi_q - \phi_r)\mathbf{x}\beta$$

Taking the exponential, we have a model that is multiplicative in the odds:

$$\Omega_{q|r}(\mathbf{x}) = \exp \{ (\theta_q - \theta_r) - (\phi_q - \phi_r)\mathbf{x}\beta \}$$

This equation can be used to estimate the factor change in the odds for a unit change in x_k , holding all other variables constant. To do this, we take the ratio of the odds after x_k increases by 1 to the odds before the change. Using basic algebra,

$$\frac{\Omega_{q|r}(\mathbf{x}, x_k + 1)}{\Omega_{q|r}(\mathbf{x}, x_k)} = \exp \{ (\phi_r - \phi_q) \beta_k \} \quad (8.7)$$

This shows that the effect of x_k on the odds of outcome q versus outcome r differs across outcome comparisons according to the difference of scaling coefficients $\phi_r - \phi_q$. As with the MNLM, we can interpret the effect of x_k on the odds as follows:

For a unit increase in x_k , the odds of outcomes q versus r change by a factor of $\exp \{ (\phi_r - \phi_q) \beta_k \}$, holding all other variables constant.

Using (8.7), we can compute the odds ratios for all pairs of outcomes. Although this formula uses three coefficients— ϕ_r , ϕ_q , and β_k —to compute the odds ratio, the identification constraints simplify computation of the odds ratios for the highest numbered category compared with the lowest numbered category (assuming that you are using the default identification assumptions and let `slogit` determine the base category). Here the base outcome is 5 so that $\phi_5 = \phi_{SR} = 0$ and $\phi_1 = \phi_{SD} = 1$. Then,

$$\begin{aligned} \frac{\Omega_{SR|SD}(\mathbf{x}, x_k + 1)}{\Omega_{SR|SD}(\mathbf{x}, x_k)} &= \exp \{ (\phi_{SD} - \phi_{SR}) \beta_k \} \\ &= \exp \{ (1 - 0) \beta_k \} \\ &= e^{\beta_k} \end{aligned}$$

Accordingly, the β 's estimated by `slogit` can be interpreted directly in terms of the odds ratio of the base outcome versus outcome 1. Although this makes it simple to examine the odds ratio for one pair of outcomes, if you stop there you can easily overlook critical aspects of your data.

The easiest way to examine the effects of each variable on the odds of all pairs of outcomes is to use `listcoef`, `expand`, where the `expand` option requests comparisons for all pairs of outcomes. Here we show the odds ratios for income:


```
. listcoef income, expand
slogit (N=1382): Factor change in odds
Odds of: StrRep vs StrDem
```

	b	z	P> z	e ^{-b}	e ^{-b} StdX	SDofX
income	0.0174	4.637	0.000	1.018	1.620	27.781
phi						
phi1_1	1.0000
phi1_2	0.6269	9.266	0.000	.	.	.
phi1_3	0.7319	9.743	0.000	.	.	.
phi1_4	0.1636	1.719	0.086	.	.	.
theta						
theta1	0.9796	2.193	0.028	.	.	.
theta2	1.4589	4.735	0.000	.	.	.
theta3	0.4512	1.282	0.200	.	.	.
theta4	0.9632	5.418	0.000	.	.	.

```
slogit (N=1382): Factor change in the odds of party
Variable: income (sd=27.781)
```

	b	z	P> z	e ^{-b}	e ^{-b} StdX
StrDem vs Dem	-0.0065	-3.424	0.001	0.994	0.835
StrDem vs Indep	-0.0047	-2.694	0.007	0.995	0.879
StrDem vs Rep	-0.0145	-4.356	0.000	0.986	0.668
StrDem vs StrRep	-0.0174	-4.637	0.000	0.983	0.617
Dem vs StrDem	0.0065	3.424	0.001	1.006	1.197
Dem vs Indep	0.0018	1.490	0.136	1.002	1.052
Dem vs Rep	-0.0080	-4.123	0.000	0.992	0.800
Dem vs StrRep	-0.0109	-4.277	0.000	0.989	0.739
Indep vs StrDem	0.0047	2.694	0.007	1.005	1.138
Indep vs Dem	-0.0018	-1.490	0.136	0.998	0.951
Indep vs Rep	-0.0099	-4.186	0.000	0.990	0.760
Indep vs StrRep	-0.0127	-4.370	0.000	0.987	0.703
Rep vs StrDem	0.0145	4.356	0.000	1.015	1.497
Rep vs Dem	0.0080	4.123	0.000	1.008	1.250
Rep vs Indep	0.0099	4.186	0.000	1.010	1.315
Rep vs StrRep	-0.0028	-1.544	0.123	0.997	0.924
StrRep vs StrDem	0.0174	4.637	0.000	1.018	1.620
StrRep vs Dem	0.0109	4.277	0.000	1.011	1.353
StrRep vs Indep	0.0127	4.370	0.000	1.013	1.423
StrRep vs Rep	0.0028	1.544	0.123	1.003	1.082

The output is similar to that produced by `slogit`. The biggest difference is that the exponentials of $\hat{\beta}_k$'s and $\hat{\beta}_k s_k$'s are shown, along with odds ratios for all comparisons. The odds ratios for income can be interpreted as follows:

For a standard deviation increase in income, about \$28,000, the odds of being a strong Democrat versus a Democrat decrease by a factor of 0.84, holding all other variables constant. The odds of being a strong Republican versus a strong Democrat increase by a factor of 1.62.

And so on for the other contrasts. Full interpretation of the odds ratios for this model is just as complicated as the MNLM with its additional parameters.

None of the odds ratios for age are significant (output not shown), reflecting that the imposed ordering of the dependent variable in the one-dimensional SLM is inconsistent with the effect of age on party affiliation. This was expected given our earlier analysis with the MNLM.

Ordinality in the one-dimensional SLM

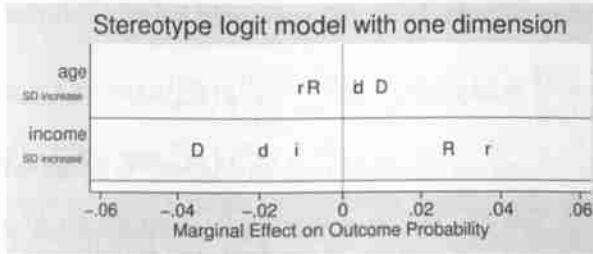
The SLM assumes that the dependent categories can be ordered, but the ordering used in fitting the model is not necessarily the same as the numbering of the outcome categories. Looking at the formula for the odds ratios,

$$\frac{\Omega_{m|n}(\mathbf{x}, x_k + 1)}{\Omega_{m|n}(\mathbf{x}, x_k)} = \exp \{ (\phi_n - \phi_m) \beta_k \}$$

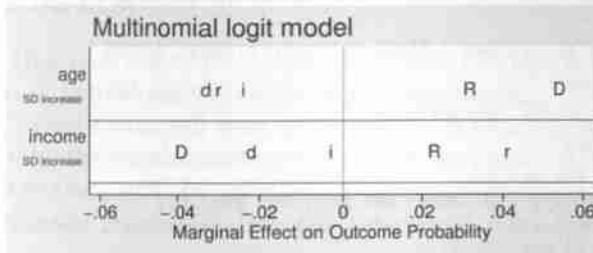
we see that the magnitude of the odds ratios will increase as category values m and n are further apart only if $\phi_1 > \phi_2 > \dots > \phi_{J-1} > \phi_J$. But `slogit` does not impose this inequality when fitting the model. If you look at the estimates from the model of party affiliation, you will see that the ϕ 's are ordered $1 = \phi_1 > \hat{\phi}_3 > \hat{\phi}_2 > \hat{\phi}_4 > \phi_5 = 0$, not $1 = \phi_1 > \hat{\phi}_2 > \hat{\phi}_3 > \hat{\phi}_4 > \phi_5 = 0$. If the ordering of categories implied by estimates of a one-dimensional SLM is not consistent with your expected ordering, this may itself prompt consideration of whether any model that assumes ordinality is appropriate.

Interpretation with predictions

The model can be interpreted using the same `m*` commands used with `ologit` or `mlogit`. For example, we can compute the average discrete changes for a standard deviation change in income and age with the command `mchange age income, amount(sd)`. Plotting the effects, we obtain



In contrast, the corresponding graph from the MNLM is



Like the OLM in chapter 7, the one-dimensional SLM imposes ordinality in a way that is inconsistent with how age increases the probability of both strong Democratic and strong Republican affiliations.

Higher-dimensional SLM

The MNLM does not require ordering of the outcome variable; the one-dimensional SLM orders the outcomes along one dimension, even if it is not the ordering that you expected. Between ordinal and fully nominal variables are variables that can be ordered on more than one dimension. For example, one dimension of party affiliation is ordered from left to right. Affiliation can also be ordered by intensity of affiliation. Ordering on multiple dimensions is possible with higher-dimensional SLMs, which we consider briefly here.

The log-linear model for a one-dimensional SLM is

$$\ln \frac{\Pr(y = q|\mathbf{x})}{\Pr(y = r|\mathbf{x})} = (\theta_q - \theta_r) - (\phi_q - \phi_r)\mathbf{x}\beta$$

For a two-dimensional model, we add another set of ϕ 's and β 's:

$$\ln \frac{\Pr(y = q|\mathbf{x})}{\Pr(y = r|\mathbf{x})} = (\theta_q - \theta_r) - (\phi_q^{[1]} - \phi_r^{[1]})\mathbf{x}\beta^{[1]} - (\phi_q^{[2]} - \phi_r^{[2]})\mathbf{x}\beta^{[2]}$$

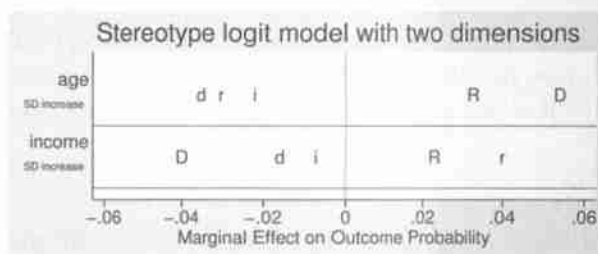
If only `age` and `income` were in the model, we would have

$$\ln \frac{\Pr(y = q|\mathbf{x})}{\Pr(y = r|\mathbf{x})} = (\theta_q - \theta_r) - (\phi_q^{[1]} - \phi_r^{[1]}) (\beta_{\text{age}}^{[1]} \text{age} + \beta_{\text{income}}^{[1]} \text{income}) \\ - (\phi_q^{[2]} - \phi_r^{[2]}) (\beta_{\text{age}}^{[2]} \text{age} + \beta_{\text{income}}^{[2]} \text{income})$$

With two dimensions, you can find that variables are significant in some, but not all, of the equations. Because the pattern of ϕ 's from the two dimensions can differ, the ordering for the first dimension (the [1] parameters) can be different from those for the second dimension (the [2] parameters).

We can fit the two-dimensional model, compute marginal effects, and plot them. The resulting graph is very similar to that from the MNLM.

```
. slogit party age income i.black i.female i.educ, dim(2)
      (output omitted)
. mchange age income, amount(sd)
      (output omitted)
. mchangeplot age income,
>   symbols(D d i r R) min(-.06) max(.06) gap(.02)
>   title("Stereotype logit model with two dimensions", position(11))
>   ysize(1.3) scale(2.1)
```



Indeed, the SLM with $J - 1$ dimensions is simply a different way to parameterize the MNLM.

8.12.2 Conditional logit model

In the MNLM, we estimate how individual-specific variables affect the likelihood of observing a specific outcome. In the conditional logit model (CLM), alternative-specific variables that vary over the possible outcomes for each individual are used to predict which outcome is chosen. In the example we will use, the outcome is the mode of transportation that an individual uses to get to work, with the possibilities being bus, car, or train. An important independent variable for transportation choice is time. Each individual has his or her own values for the amount of time it would take to get to work using each of the different modes of transportation.

In the CLM, the probability of observing outcome m is

$$\Pr(y_i = m | \mathbf{z}_i) = \frac{\exp(\mathbf{z}_{im}\gamma)}{\sum_{j=1}^J \exp(\mathbf{z}_{ij}\gamma)} \quad \text{for } m = 1 \text{ to } J$$

where \mathbf{z}_{im} contains values of the independent variables for alternative m for case i . In our example, suppose we have a single independent variable z_{im} that is the amount of time it would take respondent i to travel using a mode m of transportation, where m is either bus, car, or train. Then γ is a parameter indicating the effect of time on the probability of choosing one mode over another. In general, for each variable z_k , there are J values of the variable for each case but only the single parameter γ_k .

Data arrangement for the CLM

The CLM requires that each row of the dataset represents one alternative for one person. If we have data on N individuals who each choose from among J alternatives, then for the CLM each individual's data will span J rows, and the total dataset will have $N \times J$ rows. In our example, the variable `mode` distinguishes the modes of transportation (1 = Train, 2 = Bus, 3 = Car), and the variable `id` distinguishes different individuals. For the first two individuals,

```
. use travel4.dta, clear
(travel4.dta | Greene & Hensher 1997 mode of travel | 2014-04-01)
. list id mode choice time in 1/6, nolabel sepby(id)
```

	id	mode	choice	time
1.	1	1	0	406
2.	1	2	0	452
3.	1	3	1	180
4.	2	1	0	398
5.	2	2	0	452
6.	2	3	1	255

The variable `time` indicates the amount of travel time for each mode of transportation for each individual. The first row is for `mode` 1, indicating travel by train, for the first individual. Thus the value of `time` means that it would take this person 406 minutes to take the trip by train. The variable `choice` is 0 or 1, where 1 indicates the mode that was chosen for the trip. Both individuals above chose to travel by car, so `choice` is 1 in the rows where `mode` is 3.

Often, datasets will instead be arranged where each row represents a single individual, and each alternative-specific variable is represented as a series of variables, one for each mode. Below is the same information on two individuals that we presented before, only now arranged in this format:


```
. use travel4case.dta, clear
(travel4case.dta | Greene & Hensher 1997 1-row-per-case | 2014-04-01)
. list id choice time1 time2 time3 in 1/2, nolabel
```

	id	choice	time1	time2	time3
1.	1	3	406	452	180
2.	2	3	398	452	255

Instead of being a binary variable, **choice** now contains the value of the mode of transportation that was chosen. Meanwhile, the variables **time1**, **time2**, and **time3** represent the amount of time for each of the three modes.

We can rearrange these data so that we can fit the CLM by using the **reshape** **long** command (see [D] **reshape**). **reshape** requires us to list the stub names of the alternative-specific variables, which is **time** in the above example. We must also specify the variable that identifies unique observations with option **i(varname)** and specify the name of the new variable that indicates the different alternatives with the option **j(newvarname)**.

```
. reshape long time, i(id) j(mode)
(note: j = 1 2 3)
```

Data	wide	->	long
Number of obs.	152	->	456
Number of variables	22	->	21
j variable (3 values)		->	mode
xij variables:	time1 time2 time3	->	time

```
. list id mode choice time in 1/6, nolabel sepby(id)
```

	id	mode	choice	time
1.	1	1	3	406
2.	1	2	3	452
3.	1	3	3	180
4.	2	1	3	398
5.	2	2	3	452
6.	2	3	3	255

The results of **reshape** match the data we presented earlier, except **choice** still contains the value of the selected alternative instead of being a binary variable indicating the row of the selected alternative. We can remedy this by using the **replace** command:


```
. replace choice = (choice == mode) if choice < . & mode < .
(393 real changes made)
. list id mode choice time in 1/6, nolabel sepby(id)
```

	id	mode	choice	time
1.	1	1	0	406
2.	1	2	0	452
3.	1	3	1	180
4.	2	1	0	398
5.	2	2	0	452
6.	2	3	1	255

Fitting the CLM with `asclogit`

We can estimate the parameters of CLM by using the `asclogit` command (see [R] [asclogit](#)). `asclogit` requires that we use the options `case()` to specify the variable that identifies individuals and `alternative()` to specify the variable that identifies different alternatives. In our example, the `id` variable distinguishes individuals from one another, while the alternatives are distinguished by `mode`.

```
. use travel4.dta, clear
(travel4.dta | Greene & Hensher 1997 mode of travel | 2014-04-01)
. asclogit choice time, alt(mode) case(id) nolog

Alternative-specific conditional logit      Number of obs      =      456
Case variable: id                        Number of cases     =      152
Alternative variable: mode                Alts per case: min =        3
                                           avg =       3.0
                                           max =        3
                                           Wald chi2(1)      =      71.54
                                           Prob > chi2       =      0.0000
Log likelihood = -90.548414
```

choice	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
mode						
time	-.0200549	.0023711	-8.46	0.000	-.0247021	-.0154076
Train	(base alternative)					
Bus						
_cons	-.487722	.296565	-1.64	0.100	-1.068979	.0935347
Car						
_cons	-1.495147	.2963507	-5.05	0.000	-2.075984	-.9143105

The coefficient for `time` indicates the effect of time on the log odds that an alternative is selected. The coefficient is negative, indicating that the chances of an alternative being selected decrease as the amount of time required to travel using that alternative increases. The intercepts for `Bus` and `Car` are relative to the base alternative, which

is `Train`. By default, the base alternative is the most frequently chosen alternative; a different base alternative may be selected using the `basealternative(#)` option.

Interpreting results of the CLM

Odds ratios for the CLM can be obtained by specifying the `or` option:

```
. use travel4.dta, clear
(travel4.dta | Greene & Hensher 1997 mode of travel | 2014-04-01)
. asclogit choice time, alt(mode) case(id) nolog or
Alternative-specific conditional logit      Number of obs   =      456
Case variable: id                        Number of cases  =      152
Alternative variable: mode                Alts per case: min =       3
                                           avg   =      3.0
                                           max   =       3
                                           Wald chi2(1)    =      71.54
Log likelihood = -90.548414                Prob > chi2      =      0.0000
```

choice	Odds Ratio	Std. Err.	z	P> z	[95% Conf. Interval]	
mode						
time	.9801449	.002324	-8.46	0.000	.9756005	.9847105
Train	(base alternative)					
Bus						
_cons	.6140236	.1820979	-1.64	0.100	.343359	1.098049
Car						
_cons	.2242156	.0664464	-5.05	0.000	.125433	.4007929

The odds ratio for `time` can be interpreted as follows:

Increasing the time of travel by 1 minute for a given mode of transportation decreases the odds of using that mode by a factor of 0.98 (2%), holding the values for other alternatives constant.

Because the `margins` command is not allowed after `asclogit`, our `m*` commands do not work either. `margins` does not work because when `predict` is used after `asclogit`, it computes predictions that sum to 1 within each individual. That is, the predicted probability of a person choosing to travel by bus depends not only on how long the bus trip would take but also on how long it would take to travel by car and by train.

In place of `margins`, the command `estat mfx` can be used. `estat mfx` provides predicted probabilities and marginal effects, holding all variables at specific values. By default, `estat mfx` holds variables to their alternative-specific means. This is clearer if we look at the output:


```
. estat mfx
```

```
Pr(choice = Train|1 selected) = .46359244
```

variable		dp/dx	Std. Err.	z	P> z	[95% C.I.]	X
time							
	Train	-.004987	.000602	-8.29	0.000	-.006167 -.003808	643.44
	Bus	.001416	.000341	4.16	0.000	.000748 .002084	674.62
	Car	.003571	.000582	6.13	0.000	.00243 .004712	578.27

```
Pr(choice = Bus|1 selected) = .15232573
```

variable		dp/dx	Std. Err.	z	P> z	[95% C.I.]	X
time							
	Train	.001416	.000341	4.16	0.000	.000748 .002084	643.44
	Bus	-.00259	.000537	-4.82	0.000	-.003643 -.001536	674.62
	Car	.001173	.000287	4.09	0.000	.00061 .001736	578.27

```
Pr(choice = Car|1 selected) = .38408183
```

variable		dp/dx	Std. Err.	z	P> z	[95% C.I.]	X
time							
	Train	.003571	.000582	6.13	0.000	.00243 .004712	643.44
	Bus	.001173	.000287	4.09	0.000	.00061 .001736	674.62
	Car	-.004744	.000633	-7.50	0.000	-.005984 -.003504	578.27

The column labeled *X* at the far right contains the values at which the alternative-specific variables are held for making predictions. For alternative *Train*, *X* is 643.44, because this is the mean of *time* when *mode* is 1, indicating travel by train. The value of *X* is about a half-hour longer for *Bus*, and more than an hour shorter for *Car*, reflecting the different average times for these alternatives. The results show separate predicted probabilities for each alternative. Beneath the predicted probability for each alternative are the corresponding marginal effects. Below the probability for *Train*, for example, we have the marginal change in the probability of selecting each alternative for an increase in the time it takes to travel by *Train*. As the time it takes by train increases, the probability of traveling by train decreases, while the probabilities of traveling by bus and car increase. The marginal effects of *time* are very small because *time* is measured in minutes, while the mean trip by train takes more than 10 hours.

As with *margins*, we can use *at()* to change the values of the independent variables. If we specify *at(time=600)*, this would produce predictions and marginal effects with the time of the journey held to 600 minutes (10 hours) for each alternative, although the predictions when every alternative is held to the same value will be the same regardless of what value we use. We can also set the values to differ by alternative by specifying *at(alternativenam: variable-name=value ...)*. To give a concrete example, imagine we were interested in a particular journey that we know takes 7 hours by car, 11 by bus, and 10 by train. We can compute predicted probabilities and marginal effects as follows:


```
estat mfx, at(Car: time=420 Bus: time=660 Train: time=600)
```

```
Pr(choice = Train|1 selected) = .10557809
```

variable		dp/dx	Std. Err.	z	P> z	[95% C.I.]	x
time									
	Train	-.001894	.000408	-4.64	0.000	-.002694	-.001093		600
	Bus	.000041	.000027	1.54	0.123	-.000011	.000094		660
	Car	.001853	.000388	4.78	0.000	.001092	.002613		420

```
Pr(choice = Bus|1 selected) = .01946148
```

variable		dp/dx	Std. Err.	z	P> z	[95% C.I.]	x
time									
	Train	.000041	.000027	1.54	0.123	-.000011	.000094		600
	Bus	-.000383	.000144	-2.65	0.008	-.000665	-.0001		660
	Car	.000341	.00012	2.85	0.004	.000106	.000577		420

```
Pr(choice = Car|1 selected) = .87496043
```

variable		dp/dx	Std. Err.	z	P> z	[95% C.I.]	x
time									
	Train	.001853	.000388	4.78	0.000	.001092	.002613		600
	Bus	.000341	.00012	2.85	0.004	.000106	.000577		660
	Car	-.002194	.000452	-4.85	0.000	-.00308	-.001308		420

The predicted probability of traveling by car is now 0.87, compared with 0.11 for traveling by train, and 0.02 for traveling by bus. If we had a specific change of interest, for example, if a new railway reduced the time required to travel by train by 2 hours, we could run `estat mfx` again with the new `at()` values and compare the predicted probabilities.

Including case-specific variables in the CLM

For case-specific variables, such as an individual's income, the value of a variable does not differ across outcomes. In the MNLM with J outcomes, we estimate $J-1$ parameters for each case-specific variable. The CLM has alternative-specific variables, such as the time it takes to get to work with a given mode of transportation. For alternative-specific variables, values vary across alternatives, but we estimate one parameter for the effect of the variable.

An interesting possibility is combining the two in one model, referred to as a mixed model. For example, in explaining the choice people make on mode of transportation, we might want to know if wealthier people are more likely to drive than to take the bus. To create a mixed model, we combine the formulas for the MNLM and the CLM (see Long [1997, 178–182] and Cameron and Trivedi [2005, 500–503]):

$$\Pr(y_i = m \mid \mathbf{x}_i, \mathbf{z}_i) = \frac{\exp(\mathbf{z}_{im}\boldsymbol{\gamma} + \mathbf{x}_i\boldsymbol{\beta}_m)}{\sum_{j=1}^J \exp(\mathbf{z}_{ij}\boldsymbol{\gamma} + \mathbf{x}_i\boldsymbol{\beta}_j)} \quad \text{where } \boldsymbol{\beta}_1 = 0$$

As in the CLM, \mathbf{z}_{im} contains values of the alternative-specific variables for alternative m and case i , and γ contains the effects of the alternative-specific variables. As in the MNLM, \mathbf{x}_i contains case-specific independent variables for case i , and β_m contains coefficients for the effects on alternative m relative to the base alternative.

This mixed model can be fit using `asclogit`. Case-specific variables are specified using the `casevars()` option. To apply this to our travel example, we will add case-specific variables for household income (`hinc`) and the number of people who will be traveling together (`psize`).

```
. use travel4.dta, clear
(travel4.dta | Greene & Hensher 1997 mode of travel | 2014-04-01)
. asclogit choice time, alt(mode) case(id) casevars(hinc psize) or nolog

Alternative-specific conditional logit      Number of obs   =    456
Case variable: id                        Number of cases  =    152
Alternative variable: mode                Alts per case: min =     3
                                           avg =          3.0
                                           max =           3
                                           Wald chi2(5)    =    69.13
                                           Prob > chi2     =    0.0000

Log likelihood = -82.484583
```

choice	Odds Ratio	Std. Err.	z	P> z	[95% Conf. Interval]	
mode						
time	.9812336	.002423	-7.67	0.000	.976496	.9859941
Train	(base alternative)					
Bus						
hinc	1.040854	.0192858	2.16	0.031	1.003733	1.079348
psize	.6460673	.2304594	-1.22	0.221	.3211032	1.299903
_cons	.3771992	.2608153	-1.41	0.159	.0972759	1.462635
Car						
hinc	1.048451	.0166457	2.98	0.003	1.016329	1.081589
psize	1.427525	.39482	1.29	0.198	.8301587	2.454743
_cons	.0299135	.0226882	-4.63	0.000	.006765	.1322725

The odds ratio for time has the same interpretation as before. For `hinc` and `psize`, the odds ratios indicate the effect of an increase in the case-specific variable on the odds of selecting the alternative versus the base alternative. We could interpret as follows:

A unit increase in income increases the odds of traveling by car versus traveling by train by 4.8%, holding all else constant.

Each additional member of the traveling party decreases the odds of traveling by bus versus traveling by train by 35.4%, holding all else constant.

We have considered the CLM in the context of McFadden's choice model. In our example, the outcome is an unordered set of alternatives, where the alternatives are

the same for each individual and only one outcome is selected. The possible uses of the CLM are much broader. Many of these uses require using the `clogit` command instead of `asclogit`. Models fit using `asclogit` may be fit using `clogit`, but the data arrangement and syntax involved is more complicated. See [R] **clogit** for additional examples and references.

Fitting the MNLM using `asclogit`

If we have only case-specific variables, then the model fit by `asclogit` is equivalent to the MNLM. We think it is didactically useful for understanding the connection between the CLM and the MNLM; however, in practice, if you have only case-specific variables, you would obviously just use `mlogit` to fit the MNLM.

To fit the MNLM by using `asclogit`, we need to change our data in two ways. First, because `asclogit` does not allow factor-variable notation, we must convert `educ` into a set of dummy variables. Second, we need to use `reshape long` to arrange the data so that each row represents an alternative instead of a case. `reshape long` requires one alternative-specific variable; that is, it requires at least one set of variables whose names have the same stub followed by the numbers of the outcome categories. Because no such variable exists in our data, we generate `_tmp1` through `_tmp5` that contains all missing values and will not be used in our analysis. The `asclogit` command requires these variables even though it does not use them when fitting the MNLM.

```
. use partyid4.dta, clear
(partyid4.dta | 1992 American National Election Study | 2014-03-12)
. gen hsonly = (educ==2) if educ < .
. gen college = (educ==3) if educ < .
. gen _tmp1 = .
(1382 missing values generated)
. gen _tmp2 = .
(1382 missing values generated)
. gen _tmp3 = .
(1382 missing values generated)
. gen _tmp4 = .
(1382 missing values generated)
. gen _tmp5 = .
(1382 missing values generated)
. reshape long _tmp, i(caseid) j(partyalt)
(note: j = 1 2 3 4 5)
```

Data	wide	->	long
Number of obs.	1382	->	6910
Number of variables	22	->	19
j variable (5 values)		->	partyalt
xij variables:			
	_tmp1 _tmp2 ... _tmp5	->	_tmp

```
. gen choice = (party==partyalt) if partyalt < .
```


In the `reshape long` command above, the `j(partyalt)` option indicates that a new variable called `partyalt` should be created that contains the values of the alternatives. As before, after we use `reshape long`, we need to generate a binary variable, here named `choice`, that will equal 1 for the row that represents the chosen alternative for each individual and will equal 0 otherwise. We list the first two observations to show how the data are now arranged:

```
. list caseid choice party partyalt age female hsonly college in 1/10,
>      nolabel sepby(caseid)
```

	caseid	choice	party	partyalt	age	female	hsonly	college
1.	3001	0	4	1	31	0	0	1
2.	3001	0	4	2	31	0	0	1
3.	3001	0	4	3	31	0	0	1
4.	3001	1	4	4	31	0	0	1
5.	3001	0	4	5	31	0	0	1
6.	3002	0	5	1	89	1	0	0
7.	3002	0	5	2	89	1	0	0
8.	3002	0	5	3	89	1	0	0
9.	3002	0	5	4	89	1	0	0
10.	3002	1	5	5	89	1	0	0

For caseid 3001, the value of `party` is 4, so `choice` is 1 in its fourth row; for caseid 3002, the value of `party` is 5, and `choice` is 1 in the last row.

To fit the MNLM, we include all of our independent variables with the `casevars()` option, while `case()` identifies individuals and `alternatives()` identifies alternatives.

```
. asclogit choice, case(caseid) alternatives(partyalt)
> casevars(age income black female hsonly college) nolog

Alternative-specific conditional logit      Number of obs   =      6910
Case variable: caseid                    Number of cases   =      1382
Alternative variable: partyalt             Alts per case: min =       5
                                           avg   =      5.0
                                           max   =       5
                                           Wald chi2(24)    =     226.06
                                           Prob > chi2      =     0.0000

Log likelihood = -1960.9107
```

choice	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
1	(base alternative)					
2						
age	-.0236166	.0049609	-4.76	0.000	-.0333398	-.0138934
income	.0072787	.0040951	1.78	0.075	-.0007475	.015305
black	-.9963281	.1983914	-5.02	0.000	-1.385168	-.6074881
female	.2408435	.1668491	1.44	0.149	-.0861747	.5678617
hsonly	.3451019	.2228717	1.55	0.122	.0917187	.7819225
college	.6286363	.2952132	2.13	0.033	.050029	1.207244
_cons	1.149873	.3706106	3.10	0.002	.423489	1.876256
3						
age	-.0316178	.0066137	-4.78	0.000	-.0445803	-.0186552
income	.008498	.0051429	1.65	0.098	-.0015819	.0185779
black	-.7845096	.2570288	-3.05	0.002	-1.288277	-.2807424
female	-.1889379	.2125754	-0.89	0.374	-.6055781	.2277023
hsonly	-.0469489	.2815916	-0.17	0.868	-.5988583	.5049604
college	-.3837097	.3921017	-0.98	0.328	-1.152215	.3847955
_cons	1.043723	.4634007	2.25	0.024	.135474	1.951972
4						
age	-.0245329	.0053614	-4.58	0.000	-.0350411	-.0140247
income	.016198	.0041368	3.92	0.000	.00809	.024306
black	-2.969153	.3873236	-7.67	0.000	-3.728293	-2.210013
female	.0078597	.1774071	0.04	0.965	-.3398518	.3555712
hsonly	.3721732	.2552592	1.46	0.145	-.1281256	.872472
college	.6789539	.3225212	2.11	0.035	.046824	1.311084
_cons	.9106297	.4009588	2.27	0.023	.1247648	1.696494
5						
age	-.0028185	.00644	-0.44	0.662	-.0154407	.0098036
income	.0174695	.0045777	3.82	0.000	.0084974	.0264416
black	-3.075438	.604052	-5.09	0.000	-4.259358	-1.891518
female	-.2368373	.215026	-1.10	0.271	-.6582805	.1846059
hsonly	.5548853	.3426066	1.62	0.105	-.1166113	1.226382
college	1.374585	.3990504	3.44	0.001	.5924606	2.156709
_cons	-1.182225	.5132429	-2.30	0.021	-2.188163	-.1762875

If you compare our earlier results from `mlogit`, you will see that they are differently arranged but otherwise identical.

8.12.3 Multinomial probit model with IIA

The multinomial probit model with IIA fit by `mprobit` is the normal error counterpart to the MNLM fit by `mlogit`, in the same way that `probit` is the normal counterpart to `logit`. However, `mprobit` uses a normalization that can obscure this fact. To understand this point, we need to consider how logit and probit models can be motivated as a random utility model, in which a person maximizes his or her utility.

Let u_{im} be the utility that person i receives from alternative m . The utility is assumed to be determined by a linear combination of observed characteristics \mathbf{x}_i and random error ε_{im} :

$$u_{im} = \mathbf{x}_i \beta_m + \varepsilon_{im}$$

The utility associated with each alternative m is partly determined by chance through ε . A person chooses alternative j if the utility associated with that alternative is larger than that for any other alternative. Accordingly, the probability of alternative m being chosen is

$$\Pr(y_i = m) = \Pr(u_{im} > u_{ij} \text{ for all } j \neq m)$$

The choice that a person makes under these assumptions will not change if the utility associated with each alternative changes by some fixed amount δ . That is, if $u_{im} > u_{ij}$, then $u_{im} + \delta > u_{ij} + \delta$. The choice is based on the difference in the utilities between alternatives. We can incorporate this idea into the model by taking the difference in the utilities for two alternatives. To illustrate this, assume that there are three alternatives. We consider the utility of each alternative relative to some base alternative. It does not matter which alternative is chosen as the base, so we assume that each utility is compared with alternative 1. Accordingly, we have

$$\begin{aligned} u_{i1} - u_{i1} &= 0 \\ u_{i2} - u_{i1} &= \mathbf{x}_i(\beta_2 - \beta_1) + (\varepsilon_{i2} - \varepsilon_{i1}) \\ u_{i3} - u_{i1} &= \mathbf{x}_i(\beta_3 - \beta_1) + (\varepsilon_{i3} - \varepsilon_{i1}) \end{aligned}$$

If we define $u_{im}^* \equiv u_{im} - u_{i1}$, $\varepsilon_{im}^* \equiv \varepsilon_{im} - \varepsilon_{i1}$ and $\beta_{m|1} \equiv \beta_m - \beta_1$, the model can be written as

$$\begin{aligned} u_{i2}^* &= \mathbf{x}_i \beta_{2|1} + \varepsilon_{i2}^* \\ u_{i3}^* &= \mathbf{x}_i \beta_{3|1} + \varepsilon_{i3}^* \end{aligned}$$

The specific form of the model depends on the distribution of the errors. Assuming that the ε 's have an extreme value distribution with mean 0 and variance $\pi^2/6$ leads to the MNLM that we discussed with respect to `mlogit`. Assuming that the ε 's have a normal distribution leads to a probit-type model. To understand the model fit by `mprobit` and how it relates to the usual binary probit model, we need to pay careful attention to the assumed variance of the errors. The binary probit model fit by `probit` assumes that $\text{Var}(\varepsilon_j) = 1/2$ so that $\text{Var}(\varepsilon_2^*) = \text{Var}(\varepsilon_2) + \text{Var}(\varepsilon_1) = 1$. Because we assume that the errors are uncorrelated, $\text{Cov}(\varepsilon_1, \varepsilon_2) = 0$. Using our earlier example for labor force participation, we fit the binary probit model:


```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. probit lfp k5 k618 age i.wc i.hc lwg inc, nolog

Probit regression                                Number of obs   =       753
                                                LR chi2(7)      =      124.36
                                                Prob > chi2     =       0.0000
Log likelihood = -452.69496                    Pseudo R2       =       0.1208
```

lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
k5	-.8747111	.1135584	-7.70	0.000	-1.097281	-.6521408
k618	-.0385945	.0404893	-0.95	0.340	-.1179521	.0407631
age	-.0378235	.0076093	-4.97	0.000	-.0527375	-.0229095
wc						
college	.4883144	.1354873	3.60	0.000	.2227641	.7538647
hc						
college	.0571703	.1240053	0.46	0.645	-.1858755	.3002162
lwg	.3666287	.0877792	4.17	0.000	.1935846	.5376727
inc	-.020525	.0047769	-4.30	0.000	-.0298875	-.0111625
_cons	1.918422	.3806539	5.04	0.000	1.172354	2.66449

The coefficients are for the comparison of alternative 1 (being in the labor force) to alternative 0 (not being in the labor force), so we are estimating $\beta_{1|0}$. Using the same data with `mprobit`, we obtain

```
. mprobit lfp k5 k618 age i.wc i.hc lwg inc, nolog baseoutcome(0)

Multinomial probit regression                    Number of obs   =       753
                                                Wald chi2(7)    =      107.38
Log likelihood = -452.69496                    Prob > chi2     =       0.0000
```

lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
not_in_LF	(base outcome)					
in_LF						
k5	-1.237028	.1605958	-7.70	0.000	-1.55179	-.9222664
k618	-.0545809	.0572605	-0.95	0.340	-.1668094	.0576477
age	-.0534905	.0107612	-4.97	0.000	-.0745821	-.0323989
wc						
college	.6905808	.191608	3.60	0.000	.315036	1.066126
hc						
college	.0808511	.1753699	0.46	0.645	-.2628677	.4245699
lwg	.5170771	.1241385	4.17	0.000	.2737701	.7603841
inc	-.0290268	.0067555	-4.30	0.000	-.0422673	-.0157862
_cons	2.713059	.5383259	5.04	0.000	1.65796	3.768158

The `baseoutcome(0)` option indicates that outcome 0, not being in the labor force, is the base category, so we are estimating the coefficients $\beta_{1|0}$. If we used the `baseoutcome(1)` option, we would be estimating $\beta_{0|1}$.

Comparing the `mprobit` and `probit` output, we see that the z 's are identical, but the coefficients for `mprobit` are larger than those for `probit`. The reason is that `mprobit` assumes that $\text{Var}(\varepsilon_j) = 1$, so $\text{Var}(\varepsilon_j^*) = 2$. Or, in standard deviations, $\text{SD}(\varepsilon_j) = 1$ and $\text{SD}(\varepsilon_j^*) = \sqrt{2} \approx 1.414$. This leads to a change in scale so that the coefficients from `mprobit` will be larger by a factor of $\sqrt{2}$. For example, comparing the coefficients for `k5` for the two models, we see that $-1.237028 = \sqrt{2} \times -0.8747112$. Although this can be confusing, it does not really matter as long as you understand what `mprobit` is doing. If you compare the coefficients from `mprobit` with other analyses based on the usual `probit` model, you will be incorrect if you do not take into account the difference in scales. That is, you will incorrectly conclude that the coefficients are substantively larger in the data analyzed with `mprobit`. To avoid hand calculations to convert the coefficients from `mprobit` to the usual scale used with `probit` models, you can use the `probitparam` option to `mprobit`. For example, if we typed the command

```
mprobit lfp k5 k618 age wc hc lwg inc, nolog probitparam baseoutcome(0)
```

the estimated coefficients for `mprobit` would match those from `probit`.

As shown by Long (1997), the scale of coefficients is based on an arbitrary identification assumption that does not affect the predicted probabilities. To illustrate this, we can compare predicted probabilities for `probit` and `mprobit`. First, we compute predictions with `probit`:

```
. use binlfp4, clear
(binlfp4.dta | Mroz data on labor force participation of women | 2013-07-15)
. probit lfp k5 k618 age i.wc i.hc lwg inc, nolog
Probit regression                                Number of obs   =       753
                                                LR chi2(7)      =     124.36
                                                Prob > chi2     =     0.0000
                                                Pseudo R2      =     0.1208
Log likelihood = -452.69496
```

lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
k5	-.8747111	.1135584	-7.70	0.000	-1.097281	-.6521408
k618	-.0385945	.0404893	-0.95	0.340	-.1179521	.0407631
age	-.0378235	.0076093	-4.97	0.000	-.0527375	-.0229095
vc						
college	.4883144	.1354873	3.60	0.000	.2227641	.7538647
hc						
college	.0571703	.1240053	0.46	0.645	-.1858755	.3002162
lwg	.3656287	.0877792	4.17	0.000	.1935846	.5376727
inc	-.020525	.0047769	-4.30	0.000	-.0298875	-.0111625
_cons	1.918422	.3806539	5.04	0.000	1.172354	2.66449

```
. predict bpm_1
(option pr assumed; Pr(lfp))
```

After fitting a model with the `probit` command, `predict` computes the probability for outcome 1. Next, we use `mprobit`, where we compute predictions for both outcomes:


```
. mprobit lfp k5 k618 age i.wc i.hc lwg inc, nolog base(0)
Multinomial probit regression      Number of obs   =      753
                                   Wald chi2(7)         =     107.38
Log likelihood = -452.69496         Prob > chi2       =     0.0000
```

lfp	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
not_in_LF	(base outcome)					
in_LF						
k5	-1.237028	.1605958	-7.70	0.000	-1.55179	-.9222664
k618	-.0545809	.0572605	-0.95	0.340	-.1668094	.0576477
age	-.0534905	.0107612	-4.97	0.000	-.0745821	-.0323989
wc						
college	.6905808	.191608	3.60	0.000	.315036	1.066126
hc						
college	.0808511	.1753699	0.46	0.645	-.2628677	.4245699
lwg	.5170771	.1241385	4.17	0.000	.2737701	.7603841
inc	-.0290268	.0067555	-4.30	0.000	-.0422673	-.0157862
_cons	2.713059	.5383259	5.04	0.000	1.65796	3.768158

```
. predict mnpm_0 mnpm_1
(option pr assumed; predicted probabilities)
```

When we correlate the predictions, we find that `probit` and `mprobit` compute identical predicted probabilities:

```
. pwcorr bpm_1 mnpm_0 mnpm_1
```

	bpm_1	mnpm_0	mnpm_1
bpm_1	1.0000		
mnpm_0	-1.0000	1.0000	
mnpm_1	1.0000	-1.0000	1.0000

The model fit by `mprobit` assumes that the errors are normal. With normal errors, it is possible for the errors to be correlated across alternatives, thus potentially removing the IIA assumption (see section 8.12.4). Indeed, researchers often discuss the multinomial probit model for the case when errors are correlated, because this is the only real advantage of the multinomial probit over multinomial logit. But `mprobit` assumes that the errors are uncorrelated. Accordingly, `mprobit` fits an exact counterpart to the MNLM fit by `mlogit`—meaning that it also assumes IIA. If you use both `mprobit` and `mlogit` with the same model and data, you will get nearly identical predictions. For example, we can compare predictions by fitting both `mlogit` and `mprobit` for our model and computing the predicted probabilities of observing each outcome category. Here are the commands we use, without showing the output.


```
. use partyid4, clear
(partyid4.dta | 1992 American National Election Study | 2014-03-12)
. mprobit party age income i.black i.female i.educ, base(1)
(output omitted)
. predict mnpm_1 mnpm_2 mnpm_3 mnpm_4 mnpm_5
(option pr assumed; predicted probabilities)
. mlogit party age income i.black i.female i.educ, base(1)
(output omitted)
. predict mnml_1 mnml_2 mnml_3 mnml_4 mnml_5
(option pr assumed; predicted probabilities)
```

We then correlate the predicted probabilities for the first and second outcomes:

```
. correlate mnpm_1 mnml_1 mnpm_2 mnml_2
(obs=1382)
```

	mnpm_1	mnml_1	mnpm_2	mnml_2
mnpm_1	1.0000			
mnml_1	0.9988	1.0000		
mnpm_2	-0.0736	-0.0868	1.0000	
mnml_2	-0.0768	-0.0937	0.9939	1.0000

Clearly, there is not much difference. Train (2009, 35) points out that the thicker tails of the extreme value distribution used for the MNLM compared with the normal allow for “slightly more aberrant behavior”, but he also notes that it is unlikely this difference will be empirically distinguishable.

Finally, although the models fit by `mprobit` and `mlogit` produce nearly identical predictions, fitting models with `mprobit` takes longer because it computes integrals by Gaussian quadrature that approximates the integral by using a function computed at a limited number of evaluation or quadrature points. For our example, estimation with `mlogit` took 1 second, compared with 5.3 seconds using `mprobit`. We have not seen enough empirical consequence to justify the extra computational time required by `mprobit`. Further, probit models cannot be interpreted using odds ratios. Nonetheless, the `SPost` commands `listcoef`, `fitstat`, `mgen`, `mtable`, and `mchange` (as well as Stata’s `margins`) can be used with `mprobit`.

8.12.4 Alternative-specific multinomial probit

In section 8.12.3, we motivated the multinomial probit model for case-specific data in terms of a person choosing among alternatives to maximize utility. The person’s characteristics, such as age or education, affect the utility provided by each alternative. Here we extend the model to incorporate alternative-specific data. The inclusion of such data allows us to relax the assumption that the errors are uncorrelated, which eliminates the IIA restriction of `clogit` for alternative-specific data (see McFadden [1989]; Train [2009, part II]; Cameron and Trivedi [2005, 393–398]). The prospect of allowing correlation among errors provides the main rationale why one might fit a multinomial probit model.

In Stata, this model is referred to as the alternative-specific multinomial probit model (ASMNPM) and can be fit using `asmprobit`. The term “alternative-specific” in the name alludes to the fact that alternative-specific variables are necessary to identify the error correlations.⁵ If only case-based variables are available, the correlations are not identified. Accordingly, `asmprobit` offers a possible means of addressing the IIA problem for CLMs with alternative-specific data but not for MNLs with only case-specific data.

Formal statement of the ASMNPM

Assume that \mathbf{x}_{im} contains alternative-specific information about alternative m for case i and that ε_{im} is a random, normally distributed error. Let u_{im} be the utility that case i receives from alternative m where

$$u_{im} = \mathbf{x}_{im}\beta + \varepsilon_{im} \quad \text{for } m = 1, J$$

A person chooses alternative j when $u_{ij} > u_{im}$ for all $m \neq j$. Accordingly, with J choices, the probability of choice m is

$$\Pr(y_i = m) = \Pr(u_{im} > u_{ij} \text{ for all } j \neq m) \quad (8.8)$$

Because the errors are normally distributed, we can allow them to be correlated across the equations for different alternatives. Suppose that there are four alternatives. The covariance matrix for the ε 's would be

$$\Sigma_\varepsilon = \begin{bmatrix} \sigma_{11} & & & \\ \sigma_{21} & \sigma_{22} & & \\ \sigma_{31} & \sigma_{32} & \sigma_{33} & \\ \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_{44} \end{bmatrix}$$

If this matrix is constrained so $\Sigma_\varepsilon = \mathbf{I}$, so that the errors have a unit variance and are uncorrelated, we have the normal error counterpart to the CLM, where the errors are assumed to have an extreme value distribution. And just like the CLM, the model has the IIA property.

Although allowing the errors to be correlated relaxes the IIA condition, Train (2009, 103–114) shows that the parameters in Σ_ε are not identified unless constraints are imposed. These constraints reflect that neither adding a constant to the utility for each alternative nor dividing each utility by a constant will affect the choice that is made according to (8.8). Because $u_{im} > u_{ij}$ implies that $u_{im} + \delta > u_{ij} + \delta$, the choice of alternative m over alternative j is not affected by the base level of utility. Similarly, because $u_{im} > u_{ij}$ implies that $u_{im}\tau > u_{ij}\tau$ for all $\tau > 0$, the choice is also unaffected by the scale used to measure utility. Accordingly, we must normalize the model to eliminate the effects of the base level and scale of utility.

To remove the effect of the base level, we use the difference between each alternative's utility and the utility of the base alternative. Suppose that we select the first alternative

5. David Drukker at StataCorp was extremely helpful as we wrote this section and explored issues of identification.

as the base. The new equations specify how much utility an alternative provides beyond that provided by the first alternative:

$$\begin{aligned}u_{i1} - u_{i1} &= 0 \\u_{i2} - u_{i1} &= (\mathbf{x}_{i2} - \mathbf{x}_{i1})\beta + (\varepsilon_{i2} - \varepsilon_{i1}) \\u_{i3} - u_{i1} &= (\mathbf{x}_{i3} - \mathbf{x}_{i1})\beta + (\varepsilon_{i3} - \varepsilon_{i1}) \\u_{i4} - u_{i1} &= (\mathbf{x}_{i4} - \mathbf{x}_{i1})\beta + (\varepsilon_{i4} - \varepsilon_{i1})\end{aligned}$$

Defining $\varepsilon_{im}^* \equiv \varepsilon_{im} - \varepsilon_{i1}$, $u_{im}^* \equiv u_{im} - u_{i1}$, and $\mathbf{x}_{im}^* \equiv \mathbf{x}_{im} - \mathbf{x}_{i1}$ leads to

$$\begin{aligned}u_{i1}^* &= 0 \\u_{i2}^* &= \mathbf{x}_{i2}^*\beta + \varepsilon_{i2}^* \\u_{i3}^* &= \mathbf{x}_{i3}^*\beta + \varepsilon_{i3}^* \\u_{i4}^* &= \mathbf{x}_{i4}^*\beta + \varepsilon_{i4}^*\end{aligned}$$

By subtracting u_{i1} from each equation, we have reduced the number of errors by 1 because $\varepsilon_{i1}^* = \varepsilon_{i1} - \varepsilon_{i1} = 0$. The covariance matrix for the differenced errors is

$$\Sigma_{\varepsilon}^* = \begin{bmatrix} \sigma_{22}^* & & \\ \sigma_{32}^* & \sigma_{33}^* & \\ \sigma_{42}^* & \sigma_{43}^* & \sigma_{44}^* \end{bmatrix}$$

To set the scale, we fix the value of one of the variances σ_{mm}^* . Which variance we fix does not matter, so we arbitrarily pick σ_{22}^* . Whereas some treatments of the ASMNP fix the variance to 1, `asmpb` fixes the value to 2 (see our discussion above regarding `mpb`), which leads to

$$\Sigma_{\varepsilon}^* = \begin{bmatrix} 2 & & \\ \sigma_{32}^* & \sigma_{33}^* & \\ \sigma_{42}^* & \sigma_{43}^* & \sigma_{44}^* \end{bmatrix}$$

Fixing a base alternative and the variance of one of the differenced errors exactly identifies the regression coefficients and the covariance matrix for the differenced errors. By default, `asmpb` imposes these restrictions and estimates the parameters of Σ_{ε}^* . We discuss this further below.

Fitting the ASMNP with uncorrelated errors

If we assume that the errors are uncorrelated, the ASMNP is the normal counterpart to the CLM fit by `asclogit` or `clogit`. We will fit the model with the same independent variables and outcome as in our earlier model of transportation choice. We begin by assuming the errors are uncorrelated, so the CLM and ASMPRM are equivalent except for their assumptions about the shape and variance of the error distributions.

The ASMNP requires the same data arrangement as the CLM, in which each observation is represented using a separate row for each alternative. The options `case()`,

`alternatives()`, `casevars()`, and `basealternative()` work with `asmprobit` as they did with `asclogit`. The option `correlation()` is used to specify assumptions about the correlation structure of the errors. Because we want to assume the errors are uncorrelated, we specify the option `correlation(independent)`. The option `stddev()` specifies the variance structure of the errors. `stddev(homoskedastic)` is the counterpart to the other probit models we have considered in that the standard deviations of all errors are constrained to be equal.

```
. use travel4.dta, clear
(travel4.dta | Greene & Hensher 1997 mode of travel | 2014-04-01)
. asmprobit choice time, case(id) alternatives(mode) casevars(hinc psize)
> correlation(independent) stddev(homoskedastic) nolog

Alternative-specific multinomial probit      Number of obs      =      456
Case variable: id                          Number of cases      =      152
Alternative variable: mode                  Alts per case: min   =       3
                                              avg                 =      3.0
                                              max                 =       3

Integration sequence:      Hammersley
Integration points:        150
Log simulated-likelihood = -94.042671      Wald chi2(5)         =      86.50
                                              Prob > chi2          =      0.0000
```

choice	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
mode						
time	-.0096267	.0011214	-8.58	0.000	-.0118247	-.0074288
Train	(base alternative)					
Bus						
hinc	.0302129	.0124161	2.43	0.015	.0058778	.054548
psize	-.3849205	.252648	-1.52	0.128	-.8801015	.1102605
_cons	-.7118844	.4921849	-1.45	0.148	-1.676549	.2527803
Car						
hinc	.0406219	.0109879	3.70	0.000	.019086	.0621577
psize	.3752166	.1942277	1.93	0.053	-.0054627	.7558958
_cons	-2.552285	.4978618	-5.13	0.000	-3.528076	-1.576494

(mode=Train is the alternative normalizing location)
 (mode=Bus is the alternative normalizing scale)

After fitting the model, we can list the covariance matrix for the errors with the command `estat covariance`. This shows that the errors are uncorrelated with unit variance:

```
. estat covariance
```

	Train	Bus	Car
Train	1		
Bus	0	1	
Car	0	0	1

Comparing the coefficients between the `asclogit` and `asmprobit` models, we note that the coefficients tend to be larger for the logit model, as we would expect given its larger assumed variance, but this is not uniformly so. The p -values of coefficients deviate more from one another than we might expect.

Fitting the ASMNPM with correlated errors

The main reason to use `asmprobit` rather than the CLM is to allow correlated errors. `asmprobit` allows several ways of specifying the structure of the correlation of errors. The most general option is also the default, which can be explicitly specified as `correlation(unstructured)`.


```

. use travel4.dta, clear
(travel4.dta | Greene & Hensher 1997 mode of travel | 2014-04-01)
. asmpbchoice choice time, case(id) alternatives(mode) casevars(hinc psize) nolog
Alternative-specific multinomial probit      Number of obs      =      456
Case variable: id                          Number of cases       =      152
Alternative variable: mode                  Alts per case: min    =       3
                                           avg                  =      3.0
                                           max                  =       3

Integration sequence:      Hammersley
Integration points:        150
Log simulated-likelihood = -79.902305      Wald chi2(5)          =      28.71
                                           Prob > chi2           =      0.0000

```

choice	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
mode						
time	-.0265398	.0052347	-5.07	0.000	-.0367996	-.0162799
Train	(base alternative)					
Bus						
hinc	.0530925	.0228535	2.32	0.020	.0083004	.0978846
psize	-.2313477	.3735916	-0.62	0.536	-.9635738	.5008784
_cons	-1.516367	.8032308	-1.89	0.059	-3.09067	.0579363
Car						
hinc	.1273307	.0474833	2.68	0.007	.0342652	.2203962
psize	1.394874	.6592755	2.12	0.034	.1027177	2.68703
_cons	-9.073678	2.591035	-3.50	0.000	-14.15201	-3.995342
/lnl2_2	1.468344	.3236224	4.54	0.000	.8340557	2.102632
/l2_1	-1.465612	1.884657	-0.78	0.437	-5.159472	2.228248

(mode=Train is the alternative normalizing location)

(mode=Bus is the alternative normalizing scale)

We can once again obtain the covariance among the errors with `estat covariance`:

```
. estat covariance
```

	Bus	Car
Bus	2	
Car	-2.072688	21.00132

Note: covariances are for alternatives differenced with Train

The note indicates that the covariance matrix is for the differences in errors relative to alternative Train: $(\varepsilon_{iBus} - \varepsilon_{iTrain})$ and $(\varepsilon_{iCar} - \varepsilon_{iTrain})$. The variance for $(\varepsilon_{iBus} - \varepsilon_{iTrain})$ is fixed at 2 to identify the model, whereas the two remaining elements were estimated.

We can also obtain the estimated correlation matrix by using `estat correlation`:

```
. estat correlation
```

	Bus	Car
Bus	1.0000	
Car	-0.3198	1.0000

Note: correlations are for alternatives differenced with Train

It is tempting to use this information to describe the correlations among errors for the utilities. For example, one might incorrectly interpret the -0.32 in the output above as indicating that the errors for bus and car are inversely correlated. However, this is not the correlation of $\varepsilon_{i\text{Bus}}$ and $\varepsilon_{i\text{Car}}$, but rather of $(\varepsilon_{i\text{Bus}} - \varepsilon_{i\text{Train}})$ and $(\varepsilon_{i\text{Car}} - \varepsilon_{i\text{Train}})$, which is unlikely to be of substantive interest.

To estimate the correlation between errors, as opposed to the correlations of differences in errors, we need to use a different specification for the structure of the errors. There are different ways to do this, which are described in [R] **asmprobit**. It is essential to understand that not all the parameters in a covariance matrix of the errors can be identified without imposing additional assumptions. Regardless of how you specify the structure of Σ_ε when using the **structural** option, you must verify that all the parameters in the resulting model are identified. Indeed, Bunch and Kitamura (1990) have shown that several published articles have used probit models that were not identified. Train (2009, 100–106) illustrates how to verify identification. We do not recommend attempting to estimate the correlations among errors unless you have a clear understanding of the model and the identifying assumptions that are being imposed.

Interpretation of the estimated effects of independent variables in the ASMNPB is analogous for those described above for case-specific and alternative-specific variables in the CLM. Because ASMNPB is a probit model instead of a logit model, interpretation using odds ratios is not available. As with CLM, because the model uses alternative-specific variables, **margins** cannot be used either; instead, you can obtain predicted probabilities and marginal effects with `estat mfx`. See [R] **asmprobit postestimation** for more details.

8.12.5 Rank-ordered logit model

Some datasets record how each individual ranks each alternative. This is much more information than simply knowing which alternative is most preferred. The rank-ordered logit model (ROLM) takes advantage of this added information and can be fit with `rologit` (see [R] **rologit**). The ROLM is a generalization of the CLM for ranked outcomes (Punj and Staelin 1978; Beggs, Cardell, and Hausman 1981; Allison and Christakis 1994). As with the CLM, the ROLM can be used with case-specific explanatory variables, alternative-specific explanatory variables, or a combination of both. The model can also be used when individuals provide tied ranks or when they rank only their most preferred alternatives and leave the remainder unranked.

Perhaps the most straightforward way of understanding the ROLM is to imagine the task of ranking the alternatives as a sequence of choices. Let $y_r = m$ indicate that alternative m has rank r , that is, that alternative m is the r th choice. The probability of our first choice is $\Pr(y_1 = m_1 | \mathbf{x})$. The probability of our second choice is conditional on our first choice because if $y_1 = m_1$, then $y_2 \neq m_1$. Accordingly, the probability of the second choice is $\Pr(y_2 = m_2 | \mathbf{x}, y_1 = m_1)$. Likewise, the probability of the third choice is $\Pr(y_3 = m_3 | \mathbf{x}, y_1 = m_1, y_2 = m_2)$. If our model has only case-specific variables, then the probability of being selected first has a form that is familiar from the MNLM:

$$\Pr(y_1 = m_1 | \mathbf{x}) = \frac{\exp(\mathbf{x}\beta_{m_1|b})}{\sum_{j=1}^J \exp(\mathbf{x}\beta_{m_j|b})}$$

where \mathbf{x} contains case-specific variables, b is the base alternative, $\beta_{k,m|b}$ is the effect of x_k on the log odds of choosing alternative m over alternative b , and $\beta_{k,b|b} = 0$ for all variables k .

If alternative m_1 is ranked first, then the probability of a given outcome m_2 being ranked second is computed using a choice set that excludes m_1 . This requires that we subtract $\exp(\mathbf{x}\beta_{m_1|b})$, the term corresponding to choice m_1 , from the summation in the denominator:

$$\Pr(y_2 = m_2 | \mathbf{x}, y_1 = m_1) = \frac{\exp(\mathbf{x}\beta_{m_2|b})}{\left\{ \sum_{j=1}^J \exp(\mathbf{x}\beta_{m_j|b}) \right\} - \exp(\mathbf{x}\beta_{m_1|b})}$$

and so on.

The model is fit by maximizing the probability of observing the rank orders that were observed. We can easily extend this model to include alternative-specific variables if we expand the explanatory variables to equal $\mathbf{x}\beta_{j|b} + \mathbf{z}_j\gamma$, where \mathbf{z}_j are the values of alternative-specific variables for alternative j and γ is the vector of coefficients for the effects of the alternative-specific variables.

If you imagine rank ordering as a process of sequential choice, then talking about effects that increase the expected rank of an alternative is much the same as talking about effects that increase the risk of an alternative being selected earlier rather than later. Readers familiar with survival analysis or event history analysis might recall that certain semiparametric models—notably, the Cox proportional-hazards model—derive estimates from the rank ordering of survival times among observations (Cox 1972; Cleves et al. 2010). And, indeed, `rologit` uses the `stcox` command for survival analysis to fit the model, with individual cases in `rologit` corresponding to `strata()` in `stcox`. This is particularly handy because methods for tied survival times are well established, and tied ranks can be handled using the same logic, as nicely described for survival times by Cleves et al. (2010). Briefly stated, for tied ranks, `rologit` assumes that an ordering between the tied items exists but is not known and that the different orderings are equally likely (see [ST] `stcox` for details).

Two important points must be made about the ROLM. First, the model imposes a form of the IIA assumption. Outcomes that are close substitutes may be given similar or tied ranks, so the implications are not as dramatic as in the red bus–blue bus example described for the CLM. Nevertheless, the model still assumes that the ranking of, say, red bus versus car remains the same whether or not the blue bus is available as an alternative. Second, coefficients from the ROLM are not reversible. For example, suppose that you create two versions of a person's rank-ordering of alternatives. y_{Least} records ranks sequentially with 1 for the least preferred option, and y_{Most} records ranks with 1 for the most preferred option. The estimated coefficients of a model for y_{Least} are not the negatives of the estimates from a model of y_{Most} . Accordingly, you may wish to fit the same model with both orderings to determine how much your coding affects the results. The `reverse` option for `rologit` makes this simple to do.

Fitting the ROLM

We fit the ROLM using an example from the Wisconsin Longitudinal Study, a long-running survey of 1957 Wisconsin high school graduates (Sewell et al. 2003). In 1992, respondents were asked to rate their relative preference of a series of job characteristics, including esteem by others, variety of tasks, autonomy, and job security. The variable `jobchar` indicates the row corresponding to each alternative. The variable `rank` contains a respondent's ranks for alternatives, with 1 indicating the most preferred alternative, 2 the next most preferred, and so on. The default for `rologit` is that higher values of the outcome variable indicate more preferred alternatives, so the `reverse` option must be used.

The case-specific variables we use indicate gender (`female` = 1 for female respondents) and respondent's score on a cognitive test taken in high school (`score` is measured in standard deviations). The alternative-specific variables `high` and `low` are binary variables indicating whether respondents' current jobs are relatively high (`high` = 1) or relatively low (`low` = 1) on the attributes in question, with being moderate (neither high nor low) serving as the excluded category.

Unfortunately, `rologit` does not share the same syntax as `asclogit` and `asmpbit`, which make the inclusion of case-specific variables straightforward. Instead, to specify the intercepts for each alternative, we must use a set of indicator variables. These indicators must also be used to create interaction terms with each of the case-specific variables so that each case-specific variable will have a separate coefficient for each alternative versus the base alternative.

Fortunately, we can use factor-variable notation to create these indicator variables and interactions. In our example, we want to use security (`jobchar` = 4) as our base category, so we specify `ib4.jobchar`. The factor-variable operator to construct interactions is `##`, and we include all case-specific variables in parentheses after this operator.


```
. use wlsrank4, clear
(1992 Wisconsin Longitudinal Study data on job values)
. rologit rank high low ib4.jobchar##(c.score female),
>      group(id) reverse noomitted nolog
score 1.female omitted because of no within-id variance

Rank-ordered logistic regression      Number of obs      =      12904
Group variable: id                    Number of groups   =      3226
Ties handled via the exactm method    Obs per group: min =         4
                                      avg   =        4.00
                                      max   =         4

LR chi2(11)                          =      1947.39
Prob > chi2                          =      0.0000

Log likelihood = -6127.559
```

rank	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
high	.1780449	.0374744	4.75	0.000	.1045965	.2514934
low	-.2064148	.0425829	-4.85	0.000	-.2898758	-.1229539
jobchar						
esteem	-1.017202	.054985	-18.50	0.000	-1.12497	-.9094331
variety	.528224	.0525894	10.04	0.000	.4251506	.6312974
autonomy	-.1516741	.0510374	-2.97	0.003	-.2517055	-.0516427
jobchar#						
c.score						
esteem	.1375338	.0394282	3.49	0.000	.060256	.2148116
variety	.2590404	.0370942	6.98	0.000	.1863372	.3317437
autonomy	.2133866	.0361647	5.90	0.000	.142505	.2842682
jobchar#						
female						
esteem#fem	-.1497926	.0783025	-1.91	0.056	-.3032626	.0036774
variety#fem	-.1640212	.0728306	-2.25	0.024	-.3067666	-.0212759
autonomy#fem	-.1401769	.0718325	-1.95	0.051	-.280966	.0006123

The message *score 1.female omitted because of no within-id variance* means that no main effect for either the variable *score* or the variable *female* is estimated in this model. This is correct, because the model parameters are the interactions of these variables with the three alternatives (*esteem*, *variety*, and *autonomy*) versus the base outcome of *security*. By specifying the option *noomitted* in the *rologit* command, the estimation output is cleaner because it does not include extra rows with coefficients of 0 to represent the omitted variables.

The exponentiated coefficients from this model can be interpreted as odds ratios, as with the MNLM. Because *rologit* does not have an *or* option and our *listcoef* does not support *rologit* when factor-variable notation is used, you will need to exponentiate coefficients yourself. To give examples of interpretation, we exponentiate the coefficients for the interaction of *esteem* with both *score* and *female*.

A standard deviation increase in test score increases the odds of ranking *esteem* ahead of *security* by 14.8% ($= 100\{\exp(0.138) - 1\}$), holding other variables constant.

All else being equal, women have 13.9% ($= 100\{\exp(-0.150) - 1\}$) lower odds than men of ranking the esteem provided by a job above its security.

With the alternative-specific variable `high`, we are estimating the effect that currently having a job high on a characteristic has on a respondent reporting that he or she values that characteristic more than alternatives. For example, we estimate the effect that currently employed in a job with high autonomy versus moderate autonomy has on a respondent saying that he or she prefers autonomy over security, variety, or esteem. Because we estimate only one coefficient for `high`, the estimated effect of having a job that is high on an attribute is the same regardless of what attribute we are considering. Accordingly, the coefficient for `high` can be interpreted as follows:

All else being equal, having a job with a high level of a characteristic compared with a moderate level of a characteristic increases the odds of preferring that characteristic over an alternative by 19.5% ($= 100\{\exp(0.178) - 1\}$).

ROLM results may also be interpreted in terms of predicted probabilities of being the top-ranked alternative. However, `margins` cannot be used to generate these probabilities, nor is the `estat mfx` command available.

8.13 Conclusion

In this chapter, we considered nominal outcomes. The interpretation of models for nominal outcomes is complicated because you cannot use ordinality to simplify and organize the interpretation of results. At the same time, using a model that assumes ordinality can mask important relationships because ordinal models require that the results are consistent with an ordinal outcome, and depending on your area of application, this might not often be the case. Fortunately, estimating predictions for the MNLM is no more complicated than for the OLM; indeed, exactly the same commands can be used. In addition, of course, many outcomes that researchers study offer no pretense of being plausibly ordinal, and here the need for specific methods for treating nominal outcomes is plain.

9 Models for count outcomes

Count variables record how many times something has happened. Examples include the number of patients, hospitalizations, daily homicides, theater visits, international conflicts, beverages consumed, industrial injuries, soccer goals scored, new companies, and arrests by police. Although the linear regression model has often been applied to count outcomes, these estimates can be inconsistent or inefficient. In some cases, the linear regression model can provide reasonable results; however, it is much safer to use models specifically designed for count outcomes.

In this chapter, we consider seven regression models for count outcomes, all based on the Poisson distribution. We begin with the Poisson regression model (PRM), which is the foundation for other count models. We then consider the negative binomial regression model (NBRM), which adds unobserved, continuous heterogeneity to the PRM and often provides a much better fit to the data. To deal with outcomes where observations with zero counts are missing, we consider the zero-truncated Poisson and zero-truncated negative binomial models. By combining a zero-truncated model with a binary model, we develop the hurdle regression model, which models zero and nonzero counts in separate equations. Finally, we consider the zero-inflated Poisson and the zero-inflated negative binomial models, which assume that there are two sources of zero counts.

As with earlier chapters, we review the statistical models, consider issues of testing and fit, and then discuss methods of interpretation. These discussions are intended as a review for those who are familiar with the models. See Long (1997) for a more technical introduction to count models and Cameron and Trivedi (2013) for a definitive review. You can obtain sample do-files and datasets as explained in chapter 2.

9.1 The Poisson distribution

Because the univariate Poisson distribution is fundamental to understanding regression models for counts, we start by exploring this distribution. Let μ be the rate of occurrence or the expected number of times an event will occur during a given period of time. Let y be a random variable indicating the actual number of times an event did occur. Sometimes the event will occur fewer times than expected, even not at all, and other times it will occur more often.

The relationship between the expected count μ and the probability of observing a given count y is specified by the Poisson distribution

$$\Pr(y | \mu) = \frac{e^{-\mu} \mu^y}{y!} \quad \text{for } y = 0, 1, 2, \dots$$

where $\mu > 0$ is the sole parameter defining the distribution. $y!$ is the factorial operator; for example, $4! = 4 \times 3 \times 2 \times 1$. The easiest way to get a sense of the Poisson distribution is to compare plots of predicted probabilities for different values of the rate μ , as shown in figure 9.1.

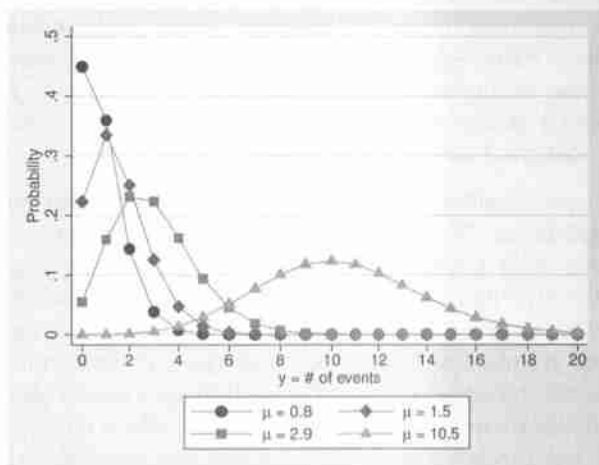


Figure 9.1. The Poisson probability density function (PDF) for different rates

The figure illustrates four characteristics of the Poisson distribution that are important for understanding regression models for counts:

1. As the mean of the distribution μ increases, the mass of the distribution shifts to the right.
2. The mean μ is also the variance. Thus $\text{Var}(y) = \mu$, which is known as equidispersion. In real data, count variables often have a variance greater than the mean, which is called overdispersion. It is possible for counts to be underdispersed, but this is rarer.
3. As μ increases, the probability of a zero count decreases rapidly. For many count variables, there are more observed 0s than predicted by the Poisson distribution.
4. As μ increases, the Poisson distribution approximates a normal distribution. This is shown by the distribution for $\mu = 10.5$.

These ideas are used as we develop regression models for count outcomes in the rest of the chapter.

Aside: Plotting the Poisson PDF. The commands below were used to create figure 9.1. The first `generate` creates variable `k`, which contains the values 0 to 20 that are the counts for which we want to compute probabilities. This is done by subtracting 1 from `_n`, where `_n` is how Stata refers to the row number of an observation. The probability of outcome k from a Poisson distribution with mean μ is computed with the function `poissonp(μ , k)` for each of four values of μ .¹

```
clear all
set obs 21
gen k = _n - 1
label var k "y = # of events"
gen psn1 = poissonp(0.8, k)
label var psn1 "&mu; = 0.8"
gen psn2 = poissonp(1.5, k)
label var psn2 "&mu; = 1.5"
gen psn3 = poissonp(2.9, k)
label var psn3 "&mu; = 2.9"
gen psn4 = poissonp(10.5, k)
label var psn4 "&mu; = 10.5"
graph twoway connected psn1 psn2 psn3 psn4 k, ///
    ytitle("Probability") ylabel(0(.1).5) xlabel(0(2)20)
    lwidth(thin thin thin thin) msymbol(0 D S T)
```

9.1.1 Fitting the Poisson distribution with the poisson command

To illustrate count models, we use data from Long (1990) on the number of articles written by biochemists in the 3 years prior to receiving their doctorate. The variables considered are

```
. use couart4, clear
(couart4.dta | Long data on Ph.D. biochemists | 2013-11-13)
. codebook art female married kid5 mentor phd, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
art	915	15	1.692896	0	19	Articles in last 3 yrs of PhD
female	915	2	.4601093	0	1	Gender: 1=female 0=male
married	915	2	.6622951	0	1	Married: 1=yes 0=no
kid5	915	4	.495082	0	3	# of kids < 6
mentor	915	49	8.767213	0	77	Mentor's # of articles
phd	915	83	3.103109	.755	4.62	PhD prestige

1. We use the Stata Markup and Control Language code `μ` to add the Greek letter μ to the labels.

The count outcome is the number of articles a scientist has published, with a distribution that is highly skewed with 30% of the cases being 0:

```
. tabulate art, missing
```

Articles in last 3 yrs of PhD	Freq.	Percent	Cum.
0	275	30.05	30.05
1	246	26.89	56.94
2	178	19.45	76.39
3	84	9.18	85.57
4	67	7.32	92.90
5	27	2.95	95.85
6	17	1.86	97.70
7	12	1.31	99.02
8	1	0.11	99.13
9	2	0.22	99.34
10	1	0.11	99.45
11	1	0.11	99.56
12	2	0.22	99.78
16	1	0.11	99.89
19	1	0.11	100.00
Total	915	100.00	

Often, the first step in analyzing a count variable is to compare the mean with the variance to determine whether there is overdispersion. By default, `summarize` does not report the variance, so we use the `detail` option:

```
. sum art, detail
```

Articles in last 3 yrs of PhD					
	Percentiles	Smallest			
1%	0	0			
5%	0	0			
10%	0	0	Obs		915
25%	0	0	Sum of Wgt.		915
50%	1		Mean		1.692896
		Largest	Std. Dev.		1.926069
75%	2	12			
90%	4	12	Variance		3.709742
95%	5	16	Skewness		2.51892
99%	7	19	Kurtosis		15.66293

The variance is more than twice as large as the mean, providing clear evidence of overdispersion.

9.1.2 Comparing observed and predicted counts with `mgen`

We can visually inspect the overdispersion in `art` by comparing the observed probabilities with those predicted from the Poisson distribution. Later, we will use the same method as a first assessment of the specification of count regression models (sec-

tion 9.2.5). We begin by using the `poisson` command to fit a model with a constant but no independent variables. When there are no independent variables, `poisson` fits a univariate Poisson distribution, where $\exp(\beta_0)$ equals the mean μ . Using `art` as the outcome,

```
. poisson art, nolog
Poisson regression
Number of obs      =       915
LR chi2(0)         =       0.00
Prob > chi2        =       .
Pseudo R2          =      0.0000
Log likelihood = -1742.5735
```

art	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
_cons	.5264408	.0254082	20.72	0.000	.4766416	.57624

Because $\hat{\beta}_0 = 0.526$, the estimated rate is $\hat{\mu} = \exp(0.526) = 1.693$, which matches the mean of `art` obtained with `summarize` earlier.

In earlier chapters, we used `mgen` to compute predictions as an independent variable changed, holding other variables constant. Although this can be done with count models, as illustrated below, the `mgen` option `meanpred` creates variables with observed and average predicted probabilities, where the rows correspond to values of the outcome.² The syntax for `mgen` used in this way is

```
mgen, meanpred stub(stub) pr(min/max) [options]
```

where option `pr(min/max)` specifies that we want to create variables with predictions for each of the counts from *min* to *max*. The new variables are

Variable name	Content
<i>stubval</i>	Value k of the dependent variable y ranging from <i>min</i> to <i>max</i> . The first row contains <i>min</i> ; the second row <i>min</i> +1; etc.
<i>stubobeq</i>	Observed proportion or probability that $y = k$. These values correspond to the percentages from <code>tabulate</code> .
<i>stuboble</i>	Observed cumulative probability that $y \leq k$.
<i>stubpreq</i>	Average predicted probability $\Pr(y = k)$.
<i>stubprle</i>	Average predicted probability $\Pr(y \leq k)$.
<i>stubobpr</i>	Difference between observed and predicted probabilities.

In a regression model with no independent variables, the predicted probability of $y = k$ is the same for all observations. Accordingly, `mgen` is simply computing $\Pr(y = k)$ from a Poisson distribution with a mean equal to the mean of the outcome. In our example,

2. `SPost9` uses the `prcounts` command to do this.


```

. poisson art, nolog
(output omitted)
. mgen, pr(0/9) meanpred stub(psn)

Predictions from:

```

Variable	Obs	Unique	Mean	Min	Max	Label
psnval	10	10	4.5	0	9	Articles in last 3 yrs...
psnobeq	10	10	.0993443	.0010929	.3005464	Observed proportion
psnoble	10	10	.8328962	.3005464	.9934427	Observed cum. proportion
psnpreq	10	10	.0999988	.0000579	.311469	Avg predicted Pr(y=#)
psnprle	10	10	.8307106	.1839859	.9999884	Avg predicted cum. Pr(...)
psnob_pr	10	10	-.0006546	-.0691068	.1165605	Observed - Avg Pr(y=#)

`mgen` created six variables with 10 observations that correspond to the counts 0–9. In the list below, `psnval` contains the count values, `psnobeq` contains observed proportions or probabilities, and `psnpreq` has predicted probabilities from a Poisson distribution with mean 1.69:

```

. list psnval psnobeq psnpreq in 1/10

```

	psnval	psnobeq	psnpreq
1.	0	.3005464	.1839859
2.	1	.2688525	.311469
3.	2	.1945355	.2636424
4.	3	.0918033	.148773
5.	4	.073224	.0629643
6.	5	.0295082	.0213184
7.	6	.0185792	.006015
8.	7	.0131148	.0014547
9.	8	.0010929	.0003078
10.	9	.0021858	.0000579

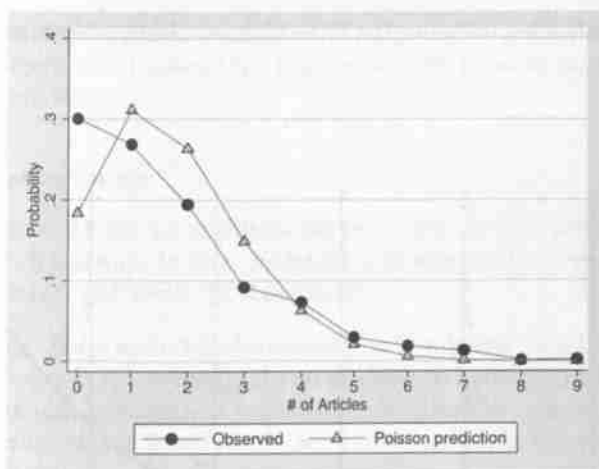
The values of `psnobeq` match those from `tabulate art` above, except that `tabulate` shows percentages while `mgen` generates probabilities, which equal the percentages divided by 100. The first row shows that the observed probability of no publications is 0.301, while the predicted probability from the Poisson distribution is only 0.184.

Using these variables, we can create a graph that compares the observed probabilities with the predicted probabilities from the Poisson distribution:

```

. label var psnobeq "Observed"
. label var psnpreq "Poisson prediction"
. label var psnval "# of Articles"
. graph twoway connected psnobeq psnpreq psnval,
> ytitle("Probability") ylabel(0(.1).4, gmax) xlabel(0/9) msym(0 Th)

```

The graph clearly shows that the fitted Poisson distribution underpredicts 0s; overpredicts counts 1, 2, and 3; and has smaller underpredictions of larger counts. This pattern of overprediction and underprediction is characteristic of count models that do not adequately account for heterogeneity among observations in their rate μ . Because the univariate Poisson distribution assumes that all scientists have exactly the same rate of productivity, which is clearly unrealistic, our next step is to allow heterogeneity in μ based on observed characteristics of the scientists.

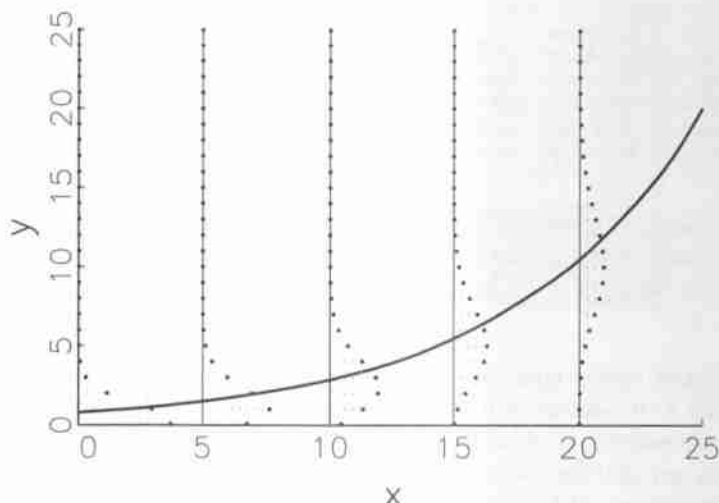
9.2 The Poisson regression model

The PRM extends the Poisson distribution by allowing each observation i to have a different rate μ_i . More formally, the PRM assumes that the observed count for observation i is drawn from a Poisson distribution with mean μ_i , where μ_i is estimated from the independent variables in the model. This is sometimes referred to as incorporating observed heterogeneity and leads to the structural equation

$$\mu_i = E(y_i | \mathbf{x}_i) = \exp(\mathbf{x}_i \boldsymbol{\beta})$$

Taking the exponential of $\mathbf{x}_i \boldsymbol{\beta}$ forces μ_i to be positive, which is necessary because counts can be only 0 or positive.

To see how this works, consider the PRM with one independent variable:



The mean $\mu = \exp(\alpha + \beta x)$ is shown by the solid, curved line that increases as x increases. For each value of μ , the Poisson distribution around the mean is shown by the dots, which should be thought of as coming out of the page to represent the probability of each count. Interpretation of the model involves assessing how changes in the independent variables affect the conditional mean and the probabilities of each count. Details on interpretation are given after we consider estimation.

9.2.1 Estimation using poisson

The PRM is fit with the command

```
poisson depvar [indepvars] [if] [in] [weight] [, noconstant
    exposure(varname) vce(vcetype) irr ]
```

In our experience, `poisson` converges quickly and difficulties are rare.

Variable lists

`depvar` is the dependent variable. `poisson` does not require this to be an integer; however, if you have noninteger values, you obtain the following warning:

```
note: you are responsible for interpretation of noncount dep. variable.
```


indepvars is a list of independent variables. If *indepvars* is not included, a model with only an intercept is estimated that fits a univariate Poisson distribution, as shown in the previous section.

Specifying the estimation sample

if and in qualifiers. *if* and *in* qualifiers can be used to restrict the estimation sample. For example, if you want to fit a model for only women, you could specify `poisson art i.mar kid5 phd ment if female==1`.

Listwise deletion. Stata excludes observations with missing values for any of the variables in the model. Accordingly, if two models are estimated using the same data but have different independent variables, it is possible to have different samples. As discussed in chapter 3, we recommend that you explicitly remove observations with missing data.

Weights and complex samples

`poisson` can be used with `fweights`, `pweights`, and `iweights`. Survey estimation for complex samples is possible using `svy`. See chapter 3 for details.

Options

`noconstant` suppresses the constant term or intercept in the model.

`exposure(varname)` specifies a variable indicating the amount of time during which an observation was “at risk” of the event occurring. Details are given in section 9.2.6.

`vce(vcetype)` specifies the type of standard errors to be computed. `vce(robust)` requests that robust variance estimates be used. See sections 3.1.9 and 9.3.5 for details.

`irr` reports estimated coefficients that are transformed to incidence-rate ratios defined as $\exp(\beta_k)$. These are discussed in section 9.2.2.

Example of the PRM

If scientists who differ in their rates of productivity are combined, the univariate distribution of articles will be overdispersed, with the variance greater than the mean. Differences among scientists in their rates of productivity could be due to factors such as quality of their graduate program, gender, marital status, number of young children, and the number of articles written by a scientist’s mentor. To account for these differences, we add these variables as independent variables:


```

. use couart4, clear
(couart4.dta | Long data on Ph.D. biochemists | 2014-04-24)
. poisson art i.female i.married kid5 phd mentor, nolog
Poisson regression
Log likelihood = -1651.0563

```

Number of obs	=	915
LR chi2(5)	=	183.03
Prob > chi2	=	0.0000
Pseudo R2	=	0.0525

art	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
female						
Female	-.2245942	.0546138	-4.11	0.000	-.3316352	-.1175532
married						
Married	.1552434	.0613747	2.53	0.011	.0349512	.2755356
kid5	-.1848827	.0401272	-4.61	0.000	-.2635305	-.1062349
phd	.0128226	.0263972	0.49	0.627	-.038915	.0645601
mentor	.0255427	.0020061	12.73	0.000	.0216109	.0294746
_cons	.3046168	.1029822	2.96	0.003	.1027755	.5064581

How you interpret a count model depends on whether you are interested in 1) the expected value or rate of the count outcome or 2) the distribution of counts. If your interest is in the rate of occurrence, several methods can be used to compute the change in the rate for a change in an independent variable, holding other variables constant. If your interest is in the distribution of counts or perhaps the probability of a specific count, such as not publishing, the probability of a count for a given level of the independent variables can be computed. We begin with interpretation using rates.

9.2.2 Factor and percentage changes in $E(y | x)$

In the PRM,

$$\mu = E(y | x) = \exp(x\beta)$$

The changes in μ as an independent variable changes can be presented in several ways. Factor change and percentage change coefficients are counterparts to odds ratios that were discussed in previous chapters. In those chapters, we expressed reservations about the usefulness of interpreting coefficients that indicated changes in the odds. As we will explain shortly, the interpretation of factor and percentage change coefficients in count models is much clearer and more useful.

Perhaps the most common method of interpretation is the factor change in the rate. Let $E(y | x, x_k)$ be the expected count for a given x , where we explicitly note the value of x_k , and let $E(y | x, x_k + 1)$ be the expected count after increasing x_k by 1. Simple algebra shows that the ratio is

$$\frac{E(y | x, x_k + 1)}{E(y | x, x_k)} = e^{\beta_k} \quad (9.1)$$

Therefore,

Factor change: For a unit change in x_k , the expected count changes by a factor of $\exp(\beta_k)$, holding other variables constant.

In some discussions of count models, μ is referred to as the incidence rate and (9.1) is called the incidence-rate ratio. These coefficients are shown by `poisson` by adding the option `irr` or by using the `listcoef` command, which is illustrated below. We can easily generalize (9.1) to changes in x_k of any amount δ :

$$\frac{E(y | \mathbf{x}, x_k + \delta)}{E(y | \mathbf{x}, x_k)} = e^{\beta_k \delta}$$

This leads to interpretations such as the following:

Factor change for δ : For a change of δ in x_k , the expected count changes by a factor of $\exp(\beta_k \delta)$, holding other variables constant.

Standardized factor change: For a standard deviation change in x_k , the expected rate changes by a factor of $\exp(\beta_k s_k)$, holding other variables constant.

Alternatively, we can compute the percentage change in the expected count for a δ -unit change in x_k , holding other variables constant:

$$100 \times \frac{E(y | \mathbf{x}, x_k + \delta) - E(y | \mathbf{x}, x_k)}{E(y | \mathbf{x}, x_k)} = 100 \times (e^{\beta_k \delta} - 1)$$

which can be interpreted as follows:

Percentage change for δ : For a change of δ in x_k , the expected count changes by $100 \times (e^{\beta_k \delta} - 1)\%$, holding other variables constant.

Whether you use percentage or factor change is a matter of taste and convention in your area of research.

Example of factor and percentage change

Factor change coefficients can be computed using `listcoef`:

```
. listcoef female mentor, help
poisson (N=915): Factor change in expected count
Observed SD: 1.9261
```

	b	z	P> z	e ^b	e ^b StdX	SDofX
female						
Female	-0.2246	-4.112	0.000	0.799	0.894	0.499
mentor	0.0255	12.733	0.000	1.026	1.274	9.484

```

b = raw coefficient
z = z-score for test of b=0
P>|z| = p-value for z-test
eb = exp(b) = factor change in expected count for unit increase in X
ebStdX = exp(b*SD of X) = change in expected count for SD increase in X
SDofX = standard deviation of X

```

By default, `listcoef` will show all coefficients, including the constant. We typed the command `listcoef female mentor` to select coefficients for only these variables. The coefficients can be interpreted as follows:

Being a female scientist decreases the expected number of articles by a factor of 0.80, holding other variables constant.

For a standard deviation increase in the mentor's productivity, roughly 9.5 articles, a scientist's expected productivity increases by a factor of 1.27, holding other variables constant.

To compute the percentage change, we add the option `percent`:

```
. listcoef female mentor, percent help
poisson (N=915): Percentage change in expected count
Observed SD: 1.9261
```

	b	z	P> z	%	%StdX	SDofX
female						
Female	-0.2246	-4.112	0.000	-20.1	-10.6	0.499
mentor	0.0255	12.733	0.000	2.6	27.4	9.484

```

b = raw coefficient
z = z-score for test of b=0
P>|z| = p-value for z-test
% = percent change in expected count for unit increase in X
%StdX = percent change in expected count for SD increase in X
SDofX = standard deviation of X

```


These can be interpreted as follows:

Being a female scientist decreases the expected number of articles by 20%, holding other variables constant.

For every additional article by the mentor, a scientist's expected productivity increases by 2.6%, holding other variables constant.

The standardized percentage change coefficient can be interpreted as follows:

For a standard deviation increase in the mentor's productivity, a scientist's expected productivity increases by 27.4%, holding other variables constant.

Factor or percentage change coefficients are quite effective for explaining the effect of variables in the PRM. Because the model is nonlinear, the specific amount of change in μ depends on the levels of all variables in the model, just like the odds in earlier chapters. However, because a multiplicative change in the rate is substantively much clearer than a multiplicative change in the odds, using factor change coefficients to interpret count models is more effective than the odds ratios we used in previous chapters. To give an example, suppose that $\exp(\beta_x) = 2$. In the PRM, we would say that for a unit increase in x , the expected number of publications doubles, holding other variables constant. If given a scientist's characteristics, her rate was $\mu = 1$, the rate becomes $\mu = 2$. If her rate was 2, it becomes 4, and so on. It is easy to understand the effect of x . If, however, we say the odds were 1 and become 2, it is not immediately obvious (for most of us) how the probability changes.

Because factor and percentage changes in the rate are an effective method for interpreting count models, we find it less important to use marginal effects. Still, they can be useful in applications when you are interested in providing a sense of the absolute amount of change in the rate. Accordingly, we consider marginal effects next.

9.2.3 Marginal effects on $E(y | \mathbf{x})$

Marginal effects indicate the change in the rate for a given change in one independent variable, holding all other variables constant. As with the models in earlier chapters, we can compute the marginal change that indicates the rate of change in $E(y | \mathbf{x})$ for an infinitely small change in x_k or a discrete change that indicates the amount of change in $E(y | \mathbf{x})$ for a discrete change in x_k .

For the PRM, the marginal change in $E(y | \mathbf{x}) = \mu$ equals

$$\frac{\partial E(y | \mathbf{x})}{\partial x_k} = E(y | \mathbf{x}) \beta_k \quad (9.2)$$

The marginal change depends on both β_k and $E(y | \mathbf{x})$. For $\beta_k > 0$, the larger the current value of $E(y | \mathbf{x})$, the larger the rate of change; for $\beta_k < 0$, the smaller the rate

of change. Because $E(y | \mathbf{x})$ depends on the levels of all variables in the model, the size of the marginal change also depends on the levels of all variables. The marginal change can be computed at the mean or at other representative values. Alternatively, the average marginal change over all the observations in the sample can be computed.

A discrete change is the change in the rate as x_k changes from x_k^{start} to x_k^{end} , while holding other variables constant:

$$\frac{\Delta E(y | \mathbf{x})}{\Delta x_k (x_k^{\text{start}} \rightarrow x_k^{\text{end}})} = E(y | \mathbf{x}, x_k = x_k^{\text{end}}) - E(y = 1 | \mathbf{x}, x_k = x_k^{\text{start}})$$

Different amounts of change can be computed depending on your purpose:

- The effect of a binary variable x_k is computed by letting x_k change from 0 to 1.
- The effect of an uncentered change of 1 in x_k is computed by changing from the observed x_{ik} to $x_{ik} + 1$. A centered change is computed by changing from $x_{ik} - (1/2)$ to $x_{ik} + (1/2)$. Change can also be computed from other values of x_k , such as the mean. Then, the uncentered unit change in x_k is from \bar{x}_k to $\bar{x}_k + 1$. The centered discrete change is the result of the change from $\bar{x}_k - (1/2)$ to $\bar{x}_k + (1/2)$.
- The effect of an uncentered change of δ , where δ might be the standard deviation of x_k , is computed by changing from x_{ik} to $x_{ik} + \delta$. The centered change is computed by changing from $x_{ik} - (\delta/2)$ to $x_{ik} + (\delta/2)$. Change can also be computed from values of x_k other than the observed value, such as the mean. Then, the uncentered change of δ in x_k is from \bar{x}_k to $\bar{x}_k + \delta$, and the centered change is from $\bar{x}_k - (\delta/2)$ to $\bar{x}_k + (\delta/2)$. Changes of a standard deviation or some other value may be particularly useful when the scale of x_k is very large or very small. For example, when the independent variable is a proportion, a unit change would be at least as large as the entire range of the variable. When the range of x_k is large, the effect of a unit change can be quite small.
- The total possible effect of x_k is found by letting x_k change from its minimum to its maximum. Trimmed ranges can also be used and may be particularly useful when the distribution of x_k is highly skewed or contains outliers.

Examples of marginal effects

Marginal effects can be computed using `mchange`. For example, the average marginal effects are

```
. poisson art i.female i.married kid5 phd mentor, nolog
(output omitted)
```

```
. mchange
```

```
poisson: Changes in mu | Number of obs = 915
```

```
Expression: Predicted number of art, predict()
```

	Change	p-value
female		
Female vs Male	-0.375	0.000
married		
Married vs Single	0.256	0.010
kid5		
+1	-0.286	0.000
+SD	-0.223	0.000
Marginal	-0.313	0.000
phd		
+1	0.022	0.629
+SD	0.022	0.629
Marginal	0.022	0.627
mentor		
+1	0.044	0.000
+SD	0.464	0.000
Marginal	0.043	0.000
Average prediction		
1.693		

Because `female` and `married` were entered using the factor-variable notation `i.female` and `i.married`, the discrete change from 0 to 1 was computed and can be interpreted as follows:

On average, for scientists similar on other characteristics, being female decreases the expected number of publications by 0.38 articles. Being married, on the other hand, increases the expected number of articles by 0.26 on average. Both effects are significant at the 0.001 level.

The most reasonable effect for the number of young children is an increase of 1 from the actual number of children a scientist has, which can be interpreted as follows:

On average, increasing the number of young children in the family by 1 decreases the expected rate of productivity by 0.29, holding other variables at their observed values.

Because the effect of doctoral prestige is not significant (given that our model includes the prestige of the mentor that is associated with the prestige of the department), we do not consider this variable further. The productivity of the mentor can be interpreted as a continuous variable. Consider the discrete change of 1:

On average, increasing the mentor's number of papers by 1 is expected to increase a scientist's productivity by 0.04 papers.

The effect is small, because an increase of one paper is tiny relative to the range of 77 for mentor's productivity. The effect of a standard deviation change gives a better sense of the effect of the mentor:

A standard deviation change in the mentor's productivity, roughly 10 papers, on average increases a scientist's expected rate of productivity by 0.46, holding other variables at their observed values.

The average marginal change could also be used to interpret the effect of the mentor's productivity. For example:

The average rate of change for the productivity of the mentor is 0.04 ($p < 0.001$). The effect of departmental prestige is not significant.

In this example, the marginal change and the discrete change of 1 for `mentor` are nearly identical, reflecting that the curve for the expected number of publications is approximately linear within a 1-unit change around the observed values.

9.2.4 Interpretation using predicted probabilities

The estimated parameters can also be used to compute predicted probabilities with the formula

$$\widehat{\Pr}(y = k \mid \mathbf{x}) = \frac{e^{-\mathbf{x}\hat{\beta}} (\mathbf{x}\hat{\beta})^k}{k!}$$

Predictions at the observed values for all observations can be made using `predict`. Probabilities at specified values or average predicted probabilities can be computed using `margins` or `mtable`. Changes in the probabilities can be computed with `mchange`, and plots can be made using `mgen`.

Predicted probabilities using `mtable` and `mchange`

To understand how independent variables affect the outcome, we can compute predicted probabilities at different levels of the variables. For example, suppose that we want to compare productivity for married and unmarried women who do not have young children, holding other variables to their global means. This can be done using `mtable`, where `width(7)` makes the output more compact:


```
. mtable, at(married=(0 1) female=1 kid5=0) atmeans pr(0/5) width(7)
Expression: Pr(art), predict(pr())
```

	married	0	1	2	3	4	5
1	0	0.244	0.344	0.243	0.114	0.040	0.011
2	1	0.193	0.317	0.261	0.143	0.059	0.019

Specified values of covariates

	female	kid5	phd	mentor
Current	1	0	3.1	8.77

Row 1 has predictions for those who are not married (that is, `married=0`), while row 2 has predictions for those who are married. Individuals who are not married have higher probabilities for zero and one publications, while married women have higher probabilities for two or more publications. The values of variables that are held constant are shown below the predictions.

We can use `mchange` to compute and test differences between women who are single and those who are married, using the option `stat(from to change p)` to request the starting prediction, the ending prediction, the change, and the *p*-value for the test that the effect is 0:

```
. mchange married, at(female=1 kid5=0) atmeans pr(0/5)
> stat(from to change p) width(7) brief
poisson: Changes in Pr(y) | Number of obs = 915
Expression: Pr(art), predict(pr())
```

		0	1	2	3	4	5
married							
	From	0.244	0.344	0.243	0.114	0.040	0.011
	To	0.193	0.317	0.261	0.143	0.059	0.019
	Married vs Single	-0.051	-0.027	0.019	0.029	0.019	0.008
	p-value	0.011	0.014	0.014	0.011	0.013	0.017

The rows `From` and `To` correspond exactly to the predictions for unmarried and married women in the earlier `mtable` output, while the row `Married vs Single` contains the discrete changes. The results show that the differences in the predicted probabilities are all statistically significant at the 0.05 level but not the 0.01 level.

In the prior example, we specified the values of all the independent variables and examined how predictions differed when the variable `married` was changed from 0 to 1. The result was a discrete change at a representative value. In our next example, we consider female scientists who are married, and we compute the average rate of productivity and average predicted probabilities if we assume these women all have no children, all have one child, all have two children, and so on. Because we are focusing on married women, we average the predictions over the subsample of married women rather than the full estimation sample. We do this by adding the condition `if married==1 & female==1` to `mtable`. We begin by computing average rates assuming different numbers of children:


```
. quietly mtable if married==1 & female==1, at(kid5=(0/3)) long
```

By default, `mtable` uses the wide format when reporting rates. Here we use the `long` option so that we can combine the rates with predicted probabilities that are by default shown in `long` format. The `norownumbers` option suppresses row numbers for the predictions.

```
. mtable if married==1 & female==1, pr(0/5)
> at(kid5=(0/3)) atvars(_none) right norownumbers brief
Expression: Pr(art), predict(pr())
```

kid5	mu	0	1	2	3	4	5
0	1.656	0.205	0.315	0.249	0.136	0.059	0.022
1	1.376	0.266	0.343	0.227	0.105	0.039	0.013
2	1.144	0.331	0.357	0.199	0.078	0.025	0.007
3	0.951	0.398	0.358	0.168	0.056	0.015	0.004

The column `mu` shows that if all married women were assumed to have no children, their average rate of productivity is 1.66 papers. Assuming all of these women have one child, the rate drops to 1.38, then 1.14 with two children, and 0.95 with three. Overall, as the number of young children increases, the rate of publication decreases by about 0.7, while the average probability of no publications increases from 0.21 to 0.40. We also see that having more young children increases the chances of having one publication and decreases the probabilities of more publications. These results are consistent with the negative effect of `kid5` in the model.

Treating a count independent variable as a factor variable

The number of young children is itself a count variable, although it is an independent variable rather than the outcome. In the PRM that we fit, the estimated coefficient for `kid5` was $\hat{\beta}_{\text{kid5}} = -0.185$, leading to the factor change in the rate of $\exp(\hat{\beta}_{\text{kid5}}) = 0.83$. This means the following:

For a unit increase in the number of children, the rate of productivity is expected to decline by a factor of 0.83, holding other variables constant.

The same rate of decline applies for the change from zero children to one child, from one child to two children, or from two to three children. It is possible, however, that the rate of change is not constant. For example, the impact of the first child could be greater than that of later children. To allow for this possibility, we can include `kid5` as a set of indicator variables by entering the variable with the factor-variable notation `i.kid5`. Fitting this model,


```
. poisson art i.kid5 i.female i.married phd mentor, nolog
Poisson regression                               Number of obs   =       915
                                                LR chi2(7)         =      184.31
                                                Prob > chi2        =       0.0000
Log likelihood = -1650.4198                    Pseudo R2         =       0.0529
```

art	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
kid5						
1	-.1786888	.0706698	-2.53	0.011	-.317199	-.0401785
2	-.3282044	.0909622	-3.61	0.000	-.5064871	-.1499217
3	-.8216372	.2816873	-2.92	0.004	-1.373734	-.2695402

(output omitted)

There are three indicator variables for the number of children, with zero children being the excluded category. We use the `mtable` commands from the last section to compute our predictions:

```
. quietly mtable if married==1 & female==1, at(kid5=(0/3)) long
. mtable if married==1 & female==1, pr(0/5) at(kid5=(0/3))
> atvars(_none) right norownumbers brief
Expression: Pr(art), predict(pr())
```

kid5	mu	0	1	2	3	4	5
0	1.648	0.207	0.316	0.249	0.135	0.059	0.022
1	1.379	0.266	0.342	0.228	0.106	0.039	0.013
2	1.187	0.318	0.355	0.205	0.083	0.027	0.008
3	0.725	0.493	0.342	0.124	0.032	0.007	0.001

There is a noticeable difference between the two models in predictions for those with three children; however, the Bayesian information criterion (BIC) statistics for the two models show very strong support for the model where `kid5` is treated as a continuous variable ($\Delta\text{BIC} = 12.36$).

In a similar vein, we might hypothesize that the effect of the mentor's productivity on a scientist's productivity decreases as the mentor's productivity increases (that is, there are diminishing returns to each additional article by the mentor). We could either use a squared term to capture this (`c.mentor##c.mentor`) or log the mentor's number of publications before including it in the model. The larger point is that while count models are nonlinear, we still need to work at specifying this nonlinearity correctly, which requires thinking not just about the left-hand side of the model but also about the relationships implied by how we specify variables on the right-hand side of the model.

Predicted probabilities using `mgen`

The `mgen` command computes a series of predictions by holding all variables but one constant (unless there are linked variables such as age and age-squared). The resulting predictions can then be plotted. To provide an example, we plot the predicted probability of not publishing for married men and married women with different numbers of children. First, we compute predictions for women by using the stub `Fprm` to indicate predictions for female scientists from the PRM:

```
. mgen if married==1 & female==1, atmeans at(kid5=(0/3))
> stub(Fprm) pr(0) predlabel(Married women)
Predictions from: margins if married==1 & female==1, atmeans at(kid5=(0/3))
> predict(pr(0))
Sample selection: if married==1 & female==1
```

Variable	Obs	Unique	Mean	Min	Max	Label
Fprmpr0	4	4	.3179806	.2011927	.4940723	Married women
Fprml10	4	4	.2333646	.1677741	.3013507	95% lower limit
Fprmul0	4	4	.4025967	.2346112	.6867938	95% upper limit
Fprmkid5	4	4	1.5	0	3	# of kids < 6
FprmCpr0	4	4	.3179806	.2011927	.4940723	pr(y<=0)


```
Specified values of covariates
```

female	married	phd	mentor
1	1	3.092822	7.88

The `predlabel()` option adds a variable label that will be used in the legend when we graph the results. Next, we compute predictions for men, using the stub `Mprm`:

```
. mgen if married==1 & female==0, atmeans at(kid5=(0/3))
> stub(Mprm) pr(0) predlabel(Married men)
Predictions from: margins if married==1 & female==0, atmeans at(kid5=(0/3))
> predict(pr(0))
Sample selection: if married==1 & female==0
```

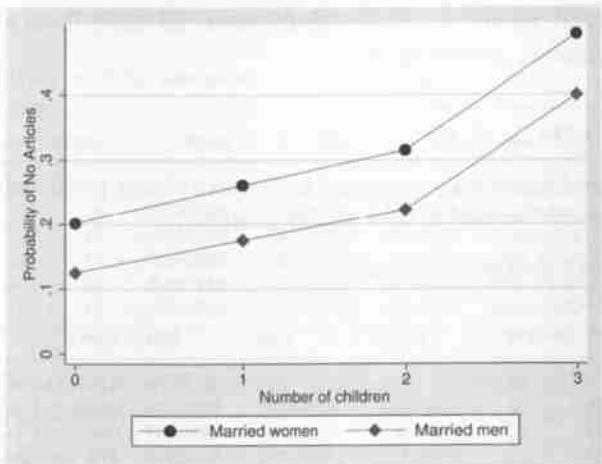
Variable	Obs	Unique	Mean	Min	Max	Label
Mprmpr0	4	4	.2309351	.1247218	.400383	Married men
Mprml10	4	4	.1531598	.0995559	.2010656	95% lower limit
Mprmul0	4	4	.3087103	.1498877	.5997005	95% upper limit
Mprmkid5	4	4	1.5	0	3	# of kids < 6
MprmCpr0	4	4	.2309351	.1247218	.400383	pr(y<=0)


```
Specified values of covariates
```

female	married	phd	mentor
0	1	3.017231	9.330709

We can then plot the predictions:

```
. graph twoway connected Fprmpr0 Mprmpr0 Mprmkid5,
> ylabel(0(.1).4, gmax) xlabel(0/3) msym(0 D)
> ytitle("Probability of No Articles") xtitle("Number of children")
```

If you compare the values plotted for women with those computed with `mtable` in the prior section, you will see that they are the same—just presented differently.

9.2.5 Comparing observed and predicted counts to evaluate model specification

Does the PRM fit the data in our example well? In this section, we present an informal way to evaluate the fit of the PRM by comparing the predicted distribution of counts with the observed distribution, extending the ideas from section 9.1.2. After fitting our model, we can compute the predicted probabilities for specific counts by using `predict`. For example, for counts of 0, 1, and 2, we type

```
. poisson art i.female i.married kid5 phd mentor, nolog
(output omitted)
. predict prob0, pr(0)
. predict prob1, pr(1)
. predict prob2, pr(2)
. sum prob0 prob1 prob2
```

Variable	Obs	Mean	Std. Dev.	Min	Max
prob0	915	.2092071	.0794247	.0000659	.4113403
prob1	915	.3098447	.0634931	.0006345	.3678775
prob2	915	.242096	.0311473	.0030544	.2706704

The means are the average predicted probabilities defined as

$$\overline{\text{Pr}}(y = k) = \frac{1}{N} \sum_{i=1}^N \widehat{\text{Pr}}(y_i = k \mid \mathbf{x}_i)$$

The average predictions from `predict` are identical to the average predictions computed by `mtable`:


```
. mtable, pr(0/2) brief
Expression: Pr(art), predict(pr())
```

	0	1	2
	0.209	0.310	0.242

The `mgen`, `meanpred` command computes the same average predicted probabilities but saves the predictions in variables that can be graphed:

```
. mgen, stub(PR) pr(0/9) meanpred
Predictions from:
```

Variable	Obs	Unique	Mean	Min	Max	Label
PRval	10	10	4.5	0	9	Articles in last 3 yrs...
PRobeq	10	10	.0993443	.0010929	.3005464	Observed proportion
PRoble	10	10	.8328962	.3005464	.9934427	Observed cum. proportion
PRpreq	10	10	.0998819	.0009304	.3098447	Avg predicted Pr(y=#)
PRprle	10	10	.8308733	.2092071	.9988188	Avg predicted cum. Pr(...)
PRob_pr	10	10	-.0005376	-.0475604	.0913393	Observed - Avg Pr(y=#)

```
. list PRval PRpreq PRobeq in 1/3, clean
```

	PRval	PRpreq	PRobeq
1.	0	.2092071	.3005464
2.	1	.3098447	.2688525
3.	2	.242096	.1945355

The listed values of `PRpreq` match those from `mtable`, and the values of `PRobeq` are the observed proportions for each value of `art`.

To show how `mgen`, `meanpred` is used to compare predictions from different models, we begin by fitting a model with no independent variables, which is simply fitting the Poisson PDF:

```
. poisson art, nolog
Poisson regression
```

Number of obs	=	915
LR chi2(0)	=	0.00
Prob > chi2	=	.
Pseudo R2	=	0.0000

```
Log likelihood = -1742.5735
```

art	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
_cons	.5264408	.0254082	20.72	0.000	.4766416 .57624

Next, we compute predictions for counts 0–9:

```
. mgen, stub(PDF) pr(0/9) meanpred
Predictions from:
```

Variable	Obs	Unique	Mean	Min	Max	Label
PDFval	10	10	4.5	0	9	Articles in last 3 yrs...
PDFobeq	10	10	.0993443	.0010929	.3005464	Observed proportion
PDFoble	10	10	.8328962	.3005464	.9934427	Observed cum. proportion
PDFpreq	10	10	.0999988	.0000579	.311469	Avg predicted Pr(y=#)
PDFprle	10	10	.8307106	.1839859	.9999884	Avg predicted cum. Pr(...)
PDFob_pr	10	10	-.0006546	-.0691068	.1165605	Observed - Avg Pr(y=#)

Because we specified the stub PDF, `mgen` created a new variable called `PDFpreq` with the average predicted probabilities for counts 0–9 from a univariate Poisson distribution. Variable `PDFobeq` contains the corresponding observed probability, while `PDFval` contains the values of the count itself (for example, 1 for the row that contains information about the observed and predicted counts of $y = 1$).

Next, we fit the PRM with the independent variables used above,

```
. poisson art i.female i.married kid5 phd mentor, nolog
(output omitted)
```

and compute the average predictions:

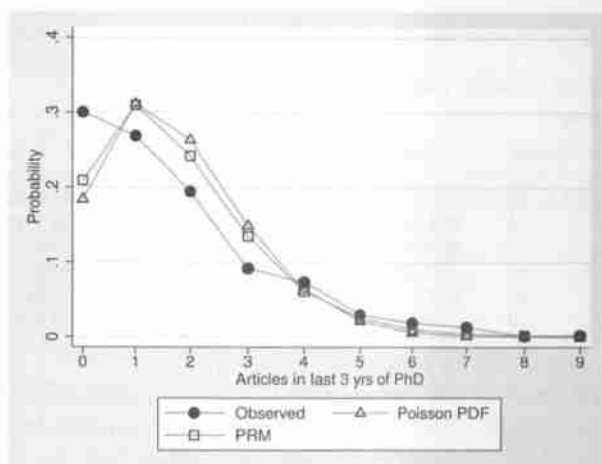
```
. mgen, stub(PRM) pr(0/9) meanpred
Predictions from:
```

Variable	Obs	Unique	Mean	Min	Max	Label
PRMval	10	10	4.5	0	9	Articles in last 3 yrs...
PRMobeq	10	10	.0993443	.0010929	.3005464	Observed proportion
PRMoble	10	10	.8328962	.3005464	.9934427	Observed cum. proportion
PRMpreq	10	10	.0998819	.0009304	.3098447	Avg predicted Pr(y=#)
PRMprle	10	10	.8308733	.2092071	.9988188	Avg predicted cum. Pr(...)
PRMob_pr	10	10	-.0005376	-.0475604	.0913393	Observed - Avg Pr(y=#)

Variable `PRMpreq` contains the average predictions based on the PRM we just fit.

We plot `PDFobeq`, `PDFpreq`, and `PRMpreq` against the count values in `PDFval` (which has the same values as `PRMval`) on the x axis:

```
. label var PDFobeq "Observed"
. label var PDFpreq "Poisson PDF"
. label var PRMpreq "PRM"
. graph twoway connected PDFobeq PDFpreq PRMpreq PRMval,
> ytitle("Probability") ylabel(0(.1).4, gmax) xlabel(0/9) msym(0 Th Sh)
```

The graph shows that, even though many of the independent variables have significant effects on the number of articles, there is only a modest improvement in the predictions made by the PRM over the univariate Poisson distribution. This suggests the need for an alternative model. Although an incorrect model might reproduce the observed distribution of counts reasonably well, systematic discrepancies between the predicted and observed distributions suggest that an alternative model should be considered. In section 9.7, we discuss other ways to compare the fit of the PRM model and alternative models.

9.2.6 (Advanced) Exposure time

This section is marked as advanced because it may be more useful to read if you are working with an application in which observations vary in terms of exposure time—that is, how long they have been at risk of the event being counted. If you have previous experience with survival models or event history models, this section may also help you better understand how modeling event counts outcomes is related to modeling whether events have happened.

So far, we have implicitly assumed that each observation was at risk of an event occurring for the same amount of time. For our example, for each person in the sample, we counted their articles over a 3-year period. Often, when collecting data, however, different observations have different exposure times. For example, the sample of scientists might have received their degrees in different years, and our outcome could have been the total publications from the PhD to the date of the survey. The amount of time in the career would clearly affect the number of publications, and scientists vary in the length of their careers. The same issue arises if we use data in which the counts

are collected from regions that have different sizes or populations. For example, if the outcome variable was the number of homicides reported in a city, the counts would be affected by the population size of the city. The methods we explain in this section could be applied in the same way.

To illustrate how to adjust for exposure time and the problems that occur if you do not make the appropriate adjustment, we have artificially constructed a variable named **profage** measuring a scientist's professional age. This is the amount of time that a scientist has been "exposed" to the possibility of publishing. The variable **totalarts** is the total number of articles during the scientist's career. Fitting a PRM, we obtain the following results:

```
. use couexposure4, clear
(couexposure4.dta | Simulated data illustrating exposure time | 2013-11-13)
. poisson totalarts kid5 mentor, nolog irr
```

Poisson regression	Number of obs	=	915
	LR chi2(2)	=	277.18
	Prob > chi2	=	0.0000
	Pseudo R2	=	0.0515
Log likelihood = -2551.3379			

totalarts	IRR	Std. Err.	z	P> z	[95% Conf. Interval]	
kid5	1.162731	.027327	6.42	0.000	1.110386	1.217544
mentor	1.024589	.0014951	16.65	0.000	1.021663	1.027524
_cons	2.132354	.0613343	26.33	0.000	2.015467	2.25602

As expected, the mentor's productivity has a strong and significant effect on the student's productivity. Surprisingly, having young children increases scientific productivity; most research in this area finds that having young children decreases productivity. One paper, however, found that having young children increases productivity. This result was an artifact from a sample where scientists with more children were older and the dependent variable was the total number of publications. Essentially, the number of children was a proxy for professional age, which has a positive effect on total publications. The same is true in our simulated data. Now, let's consider how to adjust for the artifact.

Exposure time can be easily incorporated into count models. Let t_i be the amount of time that observation i is at risk. If μ_i is the expected number of observations for one unit of time for case i , then $t_i\mu_i$ is the rate over a period of length t_i . Assuming only two independent variables for simplicity, our count equation becomes

$$\mu_i t_i = \{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2)\} \times t_i$$

Because $t = \exp(\ln t)$, the equation can be rewritten as

$$\mu_i t_i = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ln t_i)$$

This shows that the effect of different exposure times among observations can be included in the model as the log of the exposure time with its regression coefficient constrained to 1. This is exactly what Stata does with the **exposure()** option:


```
. poisson totalarts kid5 mentor, nolog irr exposure(profage)
Poisson regression
Log likelihood = -2416.7457
```

Number of obs	=	915
LR chi2(2)	=	245.43
Prob > chi2	=	0.0000
Pseudo R2	=	0.0483

totalarts	IRR	Std. Err.	z	P> z	[95% Conf. Interval]	
kid5	.8991712	.0213644	-4.47	0.000	.8582578	.9420349
mentor	1.024787	.0014781	16.98	0.000	1.021894	1.027689
_cons	1.411663	.0402961	12.08	0.000	1.334853	1.492893
ln(profage)	1	(exposure)				

Variable `ln(profage)` appears just as other variables, but the coefficient is shown as 1. There is no standard error or z statistic, because the coefficient was not estimated but was assumed to equal 1. The coefficients for `kid5` and `mentor` can be interpreted using the same methods discussed earlier. The effect of the mentor's productivity is nearly identical to our earlier results that did not adjust for exposure, but having young children now decreases scientific productivity, consistent with other studies.

We can obtain the same result by using the `offset()` option, where we specify a variable containing the log of the exposure time instead of the exposure time. For example,

```
. poisson totalarts kid5 mentor, nolog irr offset(lnprofage)
Poisson regression
Log likelihood = -2416.7457
```

Number of obs	=	915
LR chi2(2)	=	245.43
Prob > chi2	=	0.0000
Pseudo R2	=	0.0483

totalarts	IRR	Std. Err.	z	P> z	[95% Conf. Interval]	
kid5	.8991712	.0213644	-4.47	0.000	.8582578	.9420349
mentor	1.024787	.0014781	16.98	0.000	1.021894	1.027689
_cons	1.411663	.0402961	12.08	0.000	1.334853	1.492893
lnprofage	1	(offset)				

The effect of exposure time is constant over time. For those familiar with survival models, this is the same as saying the hazard of publishing is constant. To relax this assumption, we could estimate a parameter for the log of exposure time instead of constraining it to 1. This would require including `lnprofage` as an independent variable and estimating its effect just as we would any other independent variable. We could also use a different functional form to address the possibility of nonlinear effects of time on productivity.

Although the `exposure()` and `offset()` options are not considered further in this chapter, they can be used with the other models we discuss.

9.3 The negative binomial regression model

The PRM accounts for observed heterogeneity by specifying that the rate μ_i is a function of observed x_k 's. In practice, the PRM rarely fits because of overdispersion. That is, the model underfits the amount of dispersion in the outcome. The negative binomial regression model (NBRM) addresses the failure of the PRM by adding the parameter α that reflects unobserved heterogeneity among observations.³ For example, with three independent variables, the PRM is

$$\mu_i = \exp(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3})$$

The NBRM adds the error ε that is assumed to be uncorrelated with the x 's,

$$\tilde{\mu}_i = \exp(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \varepsilon_i) \quad (9.3)$$

With basic algebra and defining δ as $\exp(\varepsilon)$, the model becomes

$$\begin{aligned} \tilde{\mu}_i &= \exp(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3}) \exp(\varepsilon_i) \\ &= \exp(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3}) \delta_i \end{aligned}$$

To identify the model, we assume that $E(\delta) = 1$, which corresponds to the assumption $E(\varepsilon) = 0$ in the linear regression model. With this assumption, it follows that $E(\tilde{\mu}) = \mu E(\delta) = \mu$. Thus the PRM and the NBRM have the same mean structure. Accordingly, if the assumptions of the NBRM are correct, the expected rate for a given level of the independent variables will be the same in both models. However, the standard errors in the PRM are biased downward, resulting in spuriously large z -values and spuriously small p -values (Cameron and Trivedi 2013).⁴

To better understand the link between the PRM and the NBRM, as well as their differences, suppose that the error term ε in (9.3) is an observed variable with a regression coefficient constrained to equal 1:

$$\tilde{\mu}_i = \exp(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + 1\varepsilon_i)$$

The distribution of observations conditional on the values of both the \mathbf{x}_i and the ε_i has a Poisson distribution, just as it did for the PRM. Accordingly,

$$\Pr(y_i | \mathbf{x}_i, \varepsilon_i) = \frac{e^{-\tilde{\mu}_i} \tilde{\mu}_i^{y_i}}{y_i!}$$

However, because ε is unknown, we cannot compute $\Pr(y | \mathbf{x}, \varepsilon)$. This limitation is resolved by assuming that $\exp(\varepsilon)$ has a gamma distribution (see Long [1997, 231–232] or Cameron and Trivedi [2013, 80–89]).

3. The NBRM can also be derived through a process of contagion where the occurrence of an event changes the probability of further events—an approach not considered further here.

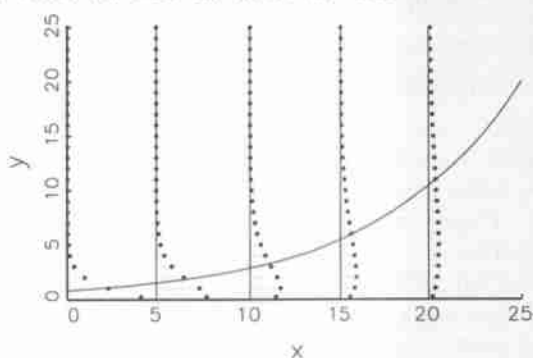
4. As we discuss in section 9.3.5, robust standard errors would be consistent and would produce p -values that yield nominal coverage, as discussed by Cameron and Trivedi (2013). Instead of following the robust approach, we focus on using a more efficient estimator whose standard errors are consistently estimated.

The probability of y conditional on \mathbf{x} , but not conditional on ε , is computed as a weighted combination of $\Pr(y | \mathbf{x}, \varepsilon)$ for all values of ε , where the weights are determined by $\Pr\{\exp(\varepsilon)\}$ from the gamma distribution. The mathematics for this mixing are complex and not particularly helpful for understanding the interpretation of the model, leading to the negative binomial distribution for y given \mathbf{x} :

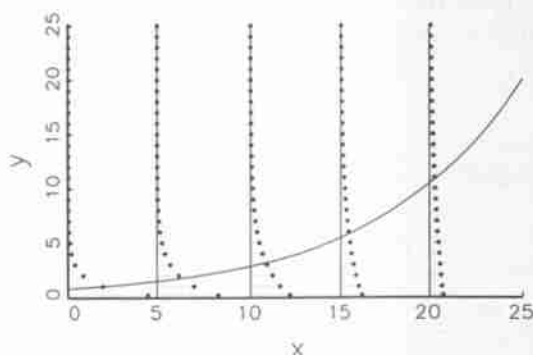
$$\Pr(y | \mathbf{x}) = \frac{\Gamma(y + \alpha^{-1})}{y! \Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu} \right)^{\alpha^{-1}} \left(\frac{\mu}{\alpha^{-1} + \mu} \right)^y$$

where $\Gamma(\cdot)$ is the gamma function. The parameter α determines the degree of dispersion in the predictions, as illustrated by the following figure:

Panel A: NBRM with $\alpha=0.5$



Panel B: NBRM with $\alpha=1.0$



In both panels, the dispersion of predicted counts for a given value of x is larger than in the PRM (compare with the figure on page 488). This is most easily seen in the greater probability of zero counts. Further, the larger value of α in panel B results in greater spread in the data. If $\alpha = 0$, the NBRM reduces to the PRM, which turns out to be the key to testing for overdispersion. This is discussed in section 9.3.3.

9.3.1 Estimation using nbreg

The NBRM is fit with the following command:

```
nbreg depvar [indepvars] [if] [in] [weight] [, noconstant
    dispersion([mean|constant]) exposure(varname) vce(vcetype) irr ]
```

Most options are the same as those for `poisson` with the notable exception of the option `dispersion()`, which is discussed in the next section. Because of differences in how `poisson` and `nbreg` are implemented in Stata, models fit with `nbreg` take longer to converge.

NB1 and NB2 variance functions

The `dispersion()` option specifies the function for the variance of y given \mathbf{x} , referred to as the dispersion function. To understand what this means, consider the dispersion function from the PRM. In that model,

$$\text{Var}(y_i | \mathbf{x}_i) = E(y_i | \mathbf{x}_i) = \mu_i$$

which is referred to as equidispersion. In real-world data, counts are usually overdispersed, meaning that the conditional variance is larger than the conditional mean. The NBRM addresses this problem by allowing overdispersion through the α parameter. Cameron and Trivedi (2013) show that a variety of variance functions are possible. The most commonly used function, the default in Stata, is

$$\text{Var}(y_i | \mathbf{x}) = \mu_i + \alpha \mu_i^2 \quad (9.4)$$

which Cameron and Trivedi refer to as the NB2 model, in reference to the squared term μ^2 . The `dispersion(constant)` option specifies the NB1 model in which

$$\text{Var}(y_i | \mathbf{x}) = \mu_i + \alpha \mu_i \quad (9.5)$$

NB1 refers to the power of 1 in the $\alpha \mu$ term. Because the NB2 model is used most often in applied research, we use this form of the model throughout the book.

9.3.2 Example of NBRM

Here we use the same example as for the PRM above:

```
. use couart4, clear
(couart4.dta | Long data on Ph.D. biochemists | 2014-04-24)
. nbreg art i.female i.married kid5 phd mentor
Fitting Poisson model:
Iteration 0:   log likelihood = -1651.4574
Iteration 1:   log likelihood = -1651.0567
Iteration 2:   log likelihood = -1651.0563
Iteration 3:   log likelihood = -1651.0563
Fitting constant-only model:
Iteration 0:   log likelihood = -1625.4242
Iteration 1:   log likelihood = -1609.9746
Iteration 2:   log likelihood = -1609.9368
Iteration 3:   log likelihood = -1609.9367
Fitting full model:
Iteration 0:   log likelihood = -1565.6652
Iteration 1:   log likelihood = -1561.0095
Iteration 2:   log likelihood = -1560.9583
Iteration 3:   log likelihood = -1560.9583
Negative binomial regression
Dispersion      = mean
Log likelihood = -1560.9583
Number of obs   =      915
LR chi2(5)      =      97.96
Prob > chi2     =      0.0000
Pseudo R2      =      0.0304
```

art	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
female						
Female	-.2164184	.0726724	-2.98	0.003	-.3588537	-.0739832
married						
Married	.1504895	.0821063	1.83	0.067	-.0104359	.3114148
kid5	-.1764152	.0530598	-3.32	0.001	-.2804105	-.07242
phd	.0152712	.0360396	0.42	0.672	-.0553652	.0859075
mentor	.0290823	.0034701	8.38	0.000	.0222811	.0358836
_cons	.256144	.1385604	1.85	0.065	-.0154294	.5277174
/lnalpha	-.8173044	.1199372			-1.052377	-.5822318
alpha	.4416205	.0529667			.3491069	.5586502

Likelihood-ratio test of alpha=0: $\chi^2_{(1)} = 180.20$ Prob>= $\chi^2 = 0.000$

The output is similar to that of `poisson` except for the results at the bottom of the output, which initially can be confusing. Although the model is defined in terms of the parameter α , `nbreg` estimates $\ln(\alpha)$, which forces the estimate of α to be positive as required for the gamma distribution. The estimate $\ln(\alpha)$ is reported as `/lnalpha`, with the value of $\hat{\alpha}$ shown on the next line. Test statistics are not given for $\ln(\alpha)$ or α because they require special treatment, as discussed next.

9.3.3 Testing for overdispersion

Because the NBRM reduces to the PRM when $\alpha = 0$, we can test for overdispersion by testing $H_0: \alpha = 0$. There are two points to remember:

1. The `nbreg` command estimates $\ln(\alpha)$ rather than α . A test of $H_0: \ln(\alpha) = 0$ corresponds to testing $H_0: \alpha = 1$, which is not the hypothesis we want to test.
2. Because α must be greater than or equal to 0, the sampling distribution of $\hat{\alpha}$ when $\alpha = 0$ is only half of a normal distribution because values less than 0 have a probability of 0. This requires an adjustment to the usual significance level of the test, which is done automatically by Stata.

Stata provides a likelihood-ratio (LR) test of $H_0: \alpha = 0$ that is listed after the estimates of the parameters:

```
Likelihood-ratio test of alpha=0:   chibar2(01) =   180.20 Prob>=chibar2 = 0.000
```

The test statistic `chibar2(01)` is computed as

$$\begin{aligned} G^2 &= 2 (\ln L_{\text{NBRM}} - \ln L_{\text{PRM}}) \\ &= 2 (-1560.9583 - -1651.0563) = 180.20 \end{aligned}$$

The log likelihood for the PRM is shown in the iteration log for `nbreg` under the heading **Fitting Poisson model**, with the log likelihood for the NBRM shown last in the log. The significance level of the test is adjusted for the truncated sampling distribution for $\hat{\alpha}$. For details, you can click on the blue linked `chibar2(01)` in the Results window. In our example, the test provides strong evidence of overdispersion. You can summarize this by saying the following:

Because there is significant evidence of overdispersion ($G^2 = 180.2$, $p < 0.001$), the NBRM is preferred over the PRM.

As with other LR tests, this test is not available if robust standard errors are used, including when probability weights, survey estimation, or the `cluster()` option is used. We talk more about robust standard errors shortly. Given that, in our experience, the test usually indicates overdispersion in cases where the LR test is available, we suggest that investigators using robust standard errors begin by fitting the NBRM—simply assume there is overdispersion.

9.3.4 Comparing the PRM and NBRM using estimates table

To understand how the PRM and NBRM differ, it is useful to compare the estimates from `poisson` and `nbreg` side by side:


```
. quietly poisson art i.female i.married kid5 phd mentor
. estimates store PRM
. quietly nbreg art i.female i.married kid5 phd mentor
. estimates store NBRM
. estimates table PRM NBRM, b(%9.3f) t p(%9.3f) varlabel
> drop(inalpha:_cons) stats(alpha N) eform vsquish
```

Variable	PRM	NBRM
female		
Female	0.799	0.805
	-4.11	-2.98
	0.000	0.003
married		
Married	1.168	1.162
	2.53	1.83
	0.011	0.067
# of kids < 6	0.831	0.838
	-4.61	-3.32
	0.000	0.001
PhD prestige	1.013	1.015
	0.49	0.42
	0.627	0.672
Mentor's # of articles	1.026	1.030
	12.73	8.38
	0.000	0.000
Constant	1.356	1.292
	2.96	1.85
	0.003	0.065
alpha		0.442
N	915	915

legend: b/t/p

The estimated parameters from the PRM and the NBRM are close, but the z -values for the NBRM are consistently smaller than those for the PRM. This is the expected consequence of overdispersion. If there is overdispersion, estimates from the PRM are inefficient, with standard errors that are biased downward (meaning that the z -values are inflated), even if the model includes the correct variables. As a consequence, if the PRM is used when there is overdispersion, the risk is that a variable will mistakenly be considered significant when it is not—as is the case for `married` in our example. Accordingly, it is important to test for overdispersion before using the PRM.

9.3.5 Robust standard errors

In the presence of overdispersion, the standard errors in the PRM are downwardly biased. Accordingly, Cameron and Trivedi (2013, 85) recommend that robust standard errors be used—not only with the PRM but also with the NBRM—in case the variance specification of the model is misspecified. To illustrate how robust standard errors can affect the statistical significance of regression coefficients, the following table compares estimates from the PRM and NBRM both with and without using robust standard errors:


```

. quietly poisson art i.female i.married kid5 phd mentor
. estimates store PRM
. quietly poisson art i.female i.married kid5 phd mentor, vce(robust)
. estimates store PRMrobust
. quietly nbreg art i.female i.married kid5 phd mentor
. estimates store NBRM
. quietly nbreg art i.female i.married kid5 phd mentor, vce(robust)
. estimates store NBRMrobust
. estimates table PRM PRMrobust NBRM NBRMrobust, b(%9.3f) se(%9.4f) p(%9.3f)
> varlabel drop(lnalpha:_cons) stats(alpha N) eform vsquish modelwidth(10)

```

Variable	PRM	PRMrobust	NBRM	NBRMrobust
female				
Female	0.799	0.799	0.805	0.805
	0.0436	0.0573	0.0585	0.0568
	0.000	0.002	0.003	0.002
married				
Married	1.168	1.168	1.162	1.162
	0.0717	0.0957	0.0954	0.0936
	0.011	0.058	0.067	0.062
# of kids < 6	0.831	0.831	0.838	0.838
	0.0334	0.0465	0.0445	0.0445
	0.000	0.001	0.001	0.001
PhD prestige	1.013	1.013	1.015	1.015
	0.0267	0.0425	0.0366	0.0381
	0.627	0.760	0.672	0.684
Mentor's # of articles	1.026	1.026	1.030	1.030
	0.0021	0.0039	0.0036	0.0040
	0.000	0.000	0.000	0.000
Constant	1.356	1.356	1.292	1.292
	0.1397	0.1988	0.1790	0.1812
	0.003	0.038	0.065	0.068
alpha			0.442	0.442
N	915	915	915	915

legend: b/se/p

There are several things to note. First, parameter estimates are not affected by using robust standard errors. For example, the coefficient for *female* is 0.799 for both PRM and PRMrobust. Second, for the PRM, the robust standard errors are substantially smaller than the nonrobust standard errors. The most noticeable difference is with *married*, where the effect is significant at the 0.01 level when robust standard errors are not used but $p = 0.06$ with robust standard errors. For the NBRM, the two types of standard errors are similar. Third, the robust standard errors in the PRM are similar to the standard errors in the NBRM, illustrating that robust standard errors for the PRM correct for the downward bias in the nonrobust standard errors. However, even if the coefficients and standard errors for the PRM estimated with robust standard errors and those for the NBRM are similar, predicted probabilities from the two models can be quite different because these probabilities depend on the dispersion parameter for the NBRM. In other words, in the presence of overdispersion, the NBRM is preferred for accurate

estimation of predicted probabilities. Cameron and Trivedi (2013, sec. 3.3) show that the NBRM is also robust to distributional misspecification when robust standard errors are used.

9.3.6 Interpretation using $E(y \mid x)$

Because the mean structure for the NBRM is identical to that for the PRM, the same methods of interpretation with rates can be used. As before, the factor change for an increase of δ in x_k equals

$$\frac{E(y \mid x, x_k + \delta)}{E(y \mid x, x_k)} = e^{\beta_k \delta}$$

and the corresponding percentage change equals

$$100 \times \frac{E(y \mid x, x_k + \delta) - E(y \mid x, x_k)}{E(y \mid x, x_k)} = 100 \times (e^{\beta_k \delta} - 1)$$

Factor and percentage change coefficients can be obtained using `listcoef`. For example, the factor change coefficients for `female` and `mentor` are

```
. nbreg art i.female i.married kid5 phd mentor, nolog
(output omitted)
. listcoef female mentor
nbreg (N=915): Factor change in expected count
Observed SD: 1.9261
```

	b	z	P> z	e ^b	e ^b StdX	SDofX
female						
Female	-0.2164	-2.978	0.003	0.805	0.898	0.499
mentor	0.0291	8.381	0.000	1.030	1.318	9.484

and the percentage change coefficients are

```
. listcoef female mentor, percent
nbreg (N=915): Percentage change in expected count
Observed SD: 1.9261
```

	b	z	P> z	%	%StdX	SDofX
female						
Female	-0.2164	-2.978	0.003	-19.5	-10.2	0.499
mentor	0.0291	8.381	0.000	3.0	31.8	9.484

These coefficients can be interpreted as follows:

Being a female scientist decreases the expected number of articles by a factor of 0.805, holding other variables constant. Equivalently, being a female scientist decreases the expected number of articles by 19.5%, holding other variables constant.

For every additional article by the mentor, a scientist's expected rate of productivity increases by 3.0%, holding other variables constant.

For a standard deviation increase in the mentor's productivity, roughly 10 articles, a scientist's expected productivity increases by 32%, holding other variables constant.

As noted earlier, factor and percentage change coefficients for count models are often quite effective for interpretation in contrast to models in previous chapters for which we think factor and percentage changes in odds are not as readily understood.

The similarity in magnitude of coefficients in the PRM and the NBRM, as well as the difference in standard errors when there is overdispersion, also affects predictions. To see this, we compute the rate of productivity for women from elite programs who have mentors with different levels of productivity:

```
. poisson art i.female i.married kid5 phd mentor, nolog
(output omitted)
. quietly mtable, estname(PRM_mu) ci at(mentor=(0(5)20) female=1 phd=4)
> atmeans clear dec(2)
. nbreg art i.female i.married kid5 phd mentor, nolog
(output omitted)
. mtable, estname(NBRM_mu) ci at(mentor=(0(5)20) female=1 phd=4)
> atmeans right dec(2) noatvars norownnumbers brief
```

Expression: Predicted number of art, predict()

mentor	PRM_mu	ll	ul	NBRM_mu	ll	ul
0	1.15	1.03	1.27	1.12	0.96	1.28
5	1.31	1.18	1.44	1.30	1.13	1.46
10	1.49	1.35	1.63	1.50	1.32	1.68
15	1.69	1.53	1.85	1.73	1.52	1.95
20	1.92	1.74	2.11	2.00	1.73	2.28

The predicted rates, in the columns PRM_mu and NBRM_mu, are similar, but the confidence intervals for the PRM are narrower than those for the NBRM, reflecting the underestimation of standard errors in the PRM when there is overdispersion. In the next section, we will show how predicted probabilities differ between the two models even when the rates are similar.

Marginal effects for the NBRM can be computed and interpreted exactly as they were for the PRM. For example, we can compute the average discrete change for increasing the number of young children in a scientist's family by 1:


```
. nbreg art i.female i.married kid5 phd mentor, nolog
(output omitted)
```

```
. mchange kid5, amount(one) stats(ci) brief
```

```
nbreg: Changes in mu | Number of obs = 915
```

```
Expression: Predicted number of art, predict()
```

		Change	LL	UL
kid5				
	+1	-0.276	-0.426	-0.125

9.3.7 Interpretation using predicted probabilities

In the presence of overdispersion, predicted probabilities from the NBRM can differ substantially from those for the PRM, even though the predicted rates are similar. The methods used to interpret predicted probabilities, however, are the same with probabilities for the NBRM computed with

$$\Pr(y | \mathbf{x}) = \frac{\Gamma(y + \alpha^{-1})}{y! \Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu} \right)^{\alpha^{-1}} \left(\frac{\mu}{\alpha^{-1} + \mu} \right)^y$$

where $\mu = \exp(\mathbf{x}\beta)$. Predicted probabilities for observed values of the independent variables can be computed using `predict`. Average predicted probabilities or predicted probabilities at specific values can be calculated using `mtable`, `mgen`, `mchange`, or `margins`. Because there is nothing new in how to use these commands with the NBRM, we provide only a few examples that are designed to illustrate key differences and similarities between the PRM and the NBRM.

In this example, we use `mtable` to compare predicted probabilities for counts 0–7, specified with the option `pr(0/7)`, for the two models for a hypothetical case of someone who is average on all characteristics.

```
. poisson art i.female i.married kid5 phd mentor, nolog
(output omitted)
```

```
. quietly mtable, rowname(PRM) pr(0/7) atmeans
```

```
. nbreg art i.female i.married kid5 phd mentor, nolog
(output omitted)
```

```
. mtable, rowname(NBRM) pr(0/7) atmeans below brief decimals(3) width(6)
```

```
Expression: Pr(art), predict(pr())
```

	0	1	2	3	4	5	6	7
PRM	0.200	0.322	0.259	0.139	0.056	0.018	0.005	0.001
NBRM	0.298	0.279	0.189	0.111	0.061	0.031	0.016	0.008

The predicted probability of a 0 is 0.20 for the PRM and 0.30 for the NBRM, a substantial difference. This difference is offset by higher probabilities for the PRM for counts 1–3. At 4 and above, the probabilities are larger for the NBRM. The larger probabilities at the higher and lower counts reflect the greater dispersion in the NBRM compared with the PRM. Differences between predictions in the NBRM and the PRM are generally most

noticeable for 0s. You can think of it this way. With overdispersion, the variance in the predicted counts increases. Because counts cannot be smaller than 0, the increased variation below the mean leads to predictions “stacking up” at 0. Above the mean, the greater variation affects all values.

To highlight the greater probability of a 0 in the NBRM, we plot the probability of 0s as mentor productivity increases from 0 to 50, holding other variables at the mean. Using `mgen`, we make predictions for the probability of 0 at different levels of the mentor’s productivity for the PRM:

```
. poisson art i.female i.married kid5 phd mentor, nolog
(output omitted)
. mgen, atmeans at(mentor=(0(5)50)) stub(PRM) pr(0)
Predictions from: margins, atmeans at(mentor=(0(5)50)) predict(pr(0))
```

Variable	Obs	Unique	Mean	Min	Max	Label
PRMpr0	11	11	.110384	.009894	.2760837	pr(y=0) from margins
PRMl10	11	11	.0954709	.0026402	.2517684	95% lower limit
PRMul0	11	11	.125297	.0171479	.3003991	95% upper limit
PRMmentor	11	11	25	0	50	Mentor's # of articles
PRMCpr0	11	11	.110384	.009894	.2760837	pr(y<=0)

```
Specified values of covariates
```

1.	1.		
female	married	kid5	phd
.4601093	.6622951	.495082	3.103109

We make corresponding predictions for the NBRM:

```
. nbreg art i.female i.married kid5 phd mentor, nolog
(output omitted)
. mgen, atmeans at(mentor=(0(5)50)) stub(NBRM) pr(0)
Predictions from: margins, atmeans at(mentor=(0(5)50)) predict(pr(0))
```

Variable	Obs	Unique	Mean	Min	Max	Label
NBRMpr0	11	11	.1954706	.0648641	.371642	pr(y=0) from margins
NBRMl10	11	11	.1629587	.0318453	.3382501	95% lower limit
NBRMul0	11	11	.2279824	.0978828	.4050339	95% upper limit
NBRMmentor	11	11	25	0	50	Mentor's # of articles
NBRMCpr0	11	11	.1954706	.0648641	.371642	pr(y<=0)

```
Specified values of covariates
```

1.	1.		
female	married	kid5	phd
.4601093	.6622951	.495082	3.103109

We use these results to graph the predictions along with their confidence intervals:

```
. label var PRMpr0 "PRM"
. label var NBRMpr0 "NBRM"
```

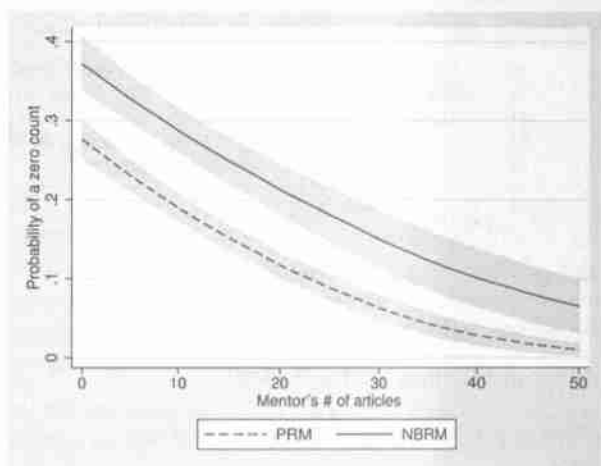


```

> . graph twoway
>   (rarea PRMl10 PRMu10 NBRMmentor, color(gs14))
>   (rarea NBRMl10 NBRMu10 NBRMmentor, color(gs14))
>   (connected PRMpr0 NBRMmentor, lpattern(dash) msize(zero))
>   (connected NBRMpr0 NBRMmentor, lpattern(solid) msize(zero)),
>   legend(on order(3 4)) ylabel(0(.1).4, gxax)
>   ylabel(0(.1).4, gxax)
>   ytitle("Probability of a zero count")

```

The `rarea` graphs add the shaded confidence intervals, with the `connected` graphs drawing the lines for predicted probabilities. The `order()` suboption of the `legend` option indicates to order the lines in the legend to start with the third graph (that is, the first `connected` graph), followed by the fourth. Because graphs 1 and 2 are not included in `order()`, no entries are included in the legend for the `rarea` graphs, which is what we want. The following graph is created:



For both models, the probability of a 0 decreases as the mentor's productivity increases, but the predicted probability of a scientist not publishing is substantially larger for the NBRM. Because both models have roughly the same expected number of publications, the higher proportion of 0s for the NBRM is being offset by the higher proportion of larger counts that are predicted by this model. The smaller confidence interval for the predictions from the PRM reflects the underestimation of standard errors when there is overdispersion. In this example, the PRM substantially underestimates the probability of a scientist not publishing, but it does this with excessive precision!

9.4 Models for truncated counts

Modeling zeros often poses special problems in count models. One type of problem arises when observations with outcomes equal to 0 are missing from the sample because of the way the data were collected. For example, suppose that we are gathering data to study scientific productivity among chemists but that we do not have a sampling roster

of all chemists with PhDs. One solution is to take a sample from those scientists who published at least one article that was listed in *Chemical Abstracts*, which excludes all chemists who have not published. Other examples of truncation are easy to find. A study of how many times people visit national parks could be based on a survey given to people entering the park. Or, when you fill out a customer satisfaction survey after buying a TV, you might be asked how many TVs are in your household, leading to a dataset in which everyone has at least one TV.

Zero-truncated count models are designed for instances like these, where observations with an outcome of 0 exist in the population but have been excluded from the sample. The zero-truncated Poisson model (ZTP) begins with the PRM from section 9.2:

$$\Pr(y_i = k | \mathbf{x}) = \frac{\exp(-\mu_i) \mu_i^k}{k!}$$

where $\mu_i = \exp(\mathbf{x}_i \beta)$. For a given \mathbf{x}_i , the probability of observing a 0 is

$$\Pr(y_i = 0 | \mathbf{x}_i) = \exp(-\mu_i)$$

and the probability of a positive (that is, nonzero) count is

$$\Pr(y_i > 0 | \mathbf{x}_i) = 1 - \exp(-\mu_i) \quad (9.6)$$

Because our counts are truncated at 0, we want to compute the probability for each positive count given that we know the count must be greater than 0 for a case to be observed. By the law of conditional probability,

$$\Pr(A | B) = \frac{\Pr(A \& B)}{\Pr(B)}$$

Thus the conditional probability of observing a specific value $y = k$ given that we know the count is not 0 is

$$\Pr(y_i = k | y_i > 0, \mathbf{x}_i) = \frac{\Pr(y_i = k \& y_i > 0 | \mathbf{x}_i)}{\Pr(y_i > 0 | \mathbf{x}_i)}$$

Given the probability that $y = k$ and $y > 0$ is simply the probability that $y = k$, and substituting (9.6) leads to the conditional probability

$$\Pr(y_i = k | y_i > 0, \mathbf{x}_i) = \frac{\Pr(y_i = k | \mathbf{x}_i)}{1 - \exp(-\mu_i)} \quad (9.7)$$

This formula increases each unconditional probability by the factor $\{1 - \exp(-\mu)\}^{-1}$, forcing the probability mass of the truncated distribution to sum to 1.

With a truncated model, there are two types of expected counts that might be of interest. First, we can compute the expected rate without truncation. For example, we might want to estimate the expected number of publications for scientists in the entire population, not just among those who have at least one article. We will refer to this

as the unconditional prediction. Importantly, we use “unconditional” to indicate that the prediction applies to the whole population; unconditional predictions are still, of course, conditional on \mathbf{x} . In the ZTP, the unconditional rate has the same formula as for the standard PRM:

$$E(y_i | \mathbf{x}_i) = \exp(\mathbf{x}_i \boldsymbol{\beta})$$

Second, we can compute the expected rate given that the count is positive, written as $E(y | y > 0, \mathbf{x})$. For example, among those who publish, what is the expected number of publications? Or, among those who see the doctor at least once a year, what is the average number of visits with a doctor? The conditional rate must be larger than the unconditional rate because we are excluding 0s. As with the conditional probability in (9.7), the rate increases proportionally to the probability of a positive count:

$$E(y_i | y_i > 0, \mathbf{x}_i) = \frac{\mu_i}{\Pr(y_i > 0 | \mathbf{x}_i)} = \frac{\mu_i}{1 - \exp(-\mu_i)} \quad (9.8)$$

The same idea applies to the NBRM, where

$$\Pr(y_i | \mathbf{x}_i) = \frac{\Gamma(y_i + \alpha^{-1})}{y_i! \Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu_i} \right)^{\alpha^{-1}} \left(\frac{\mu_i}{\alpha^{-1} + \mu_i} \right)^{y_i}$$

Accordingly, $\Pr(y = 0 | \mathbf{x}) = (1 + \alpha\mu)^{-1/\alpha}$ and $\Pr(y > 0 | \mathbf{x}) = 1 - (1 + \alpha\mu)^{-1/\alpha}$. The conditional probability in the zero-truncated negative binomial model (ZTNB) is

$$\Pr(y_i | y_i > 0, \mathbf{x}_i) = \frac{\Pr(y_i | \mathbf{x}_i)}{1 - (1 + \alpha\mu_i)^{-1/\alpha}}$$

and the conditional mean equals

$$E(y_i | y_i > 0, \mathbf{x}_i) = \frac{\mu_i}{1 - (1 + \alpha\mu_i)^{-1/\alpha}} \quad (9.9)$$

The adverse effects of overdispersion are worse with truncated models. When the sample is not truncated, using the PRM in the presence of overdispersion underestimates the standard errors but does not bias the estimated β 's. With the ZTP, overdispersion results in biased and inconsistent estimates of the β 's and, consequently, in biased estimates of rates and probabilities (Grogger and Carson 1991). Accordingly, before using the ZTP, you should check for overdispersion by fitting the ZTNB. As with the NBRM, overdispersion in the ZTNB is based on an LR test of $\alpha = 0$, which is included in the output for the ZTNB (see page 511).

9.4.1 Estimation using `tpoisson` and `tnbreg`

The ZTP is fit with the command `tpoisson`, and the ZTNB is fit with the command `tnbreg`:

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

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

where *options* are largely the same as those for `poisson` and `nbreg`. These commands can also be used when the sample is truncated at a value greater than 0. Imagine a study in which the outcome is number of criminal offenses with data that are collected only on people with multiple offenses, so only people with a count of at least two are included in the dataset. A model for this outcome can be fit by adding the `ll(#)` option to either `tpoisson` or `tnbreg`, where `#` is the value of the largest nonobserved count. In the example where only individuals with a count of at least two are included, this option should be specified as `ll(1)`. The default for both `tpoisson` and `tnbreg` is a zero-truncated model (that is, `ll(0)`), so you do not have to specify this option for zero-truncated models.⁵

Example of zero-truncated model

To illustrate zero-truncated count models, we continue our example of scientific productivity with the outcome variable `arttrunc`, which artificially recodes values of 0 in `art` to missing. We begin by truncating the sample to exclude scientists who have no publications.

```
. use couart4, clear
(couart4.dta | Long data on Ph.D. biochemists | 2014-04-24)
. sum art arttrunc
```

Variable	Obs	Mean	Std. Dev.	Min	Max
art	915	1.692896	1.926069	0	19
arttrunc	640	2.420313	1.882269	1	19

```
. drop if art==0
(275 observations deleted)
. sum art arttrunc
```

Variable	Obs	Mean	Std. Dev.	Min	Max
art	640	2.420313	1.882269	1	19
arttrunc	640	2.420313	1.882269	1	19

After dropping those 275 observations, we have a truncated sample of 640 scientists.

5. In Stata 12, `tpoisson` and `tnbreg` replaced the `ztp` and `ztnb` commands that only allow truncation at 0. Users of Stata 11 or before should use the latter commands. Because `predict` does not support the prediction of specific counts or ranges when `ztp` or `ztnb` are used, quantities based on predicted probabilities cannot be computed using `margins` or our `SPost` commands.

Next, we fit the ZTNB with the truncated sample, where we used the `irr` option so that the parameters are shown as factor change coefficients:

```
. tnbreg arttrunc i.female i.married kid5 phd mentor, nolog irr
Truncated negative binomial regression      Number of obs   =      640
Truncation point: 0                        LR chi2(5)       =      44.58
Dispersion = mean                          Prob > chi2      =      0.0000
Log likelihood = -1027.3185                 Pseudo R2       =      0.0212
```

arttrunc	IRR	Std. Err.	z	P> z	[95% Conf. Interval]	
female						
Female	.7829619	.0761181	-2.52	0.012	.6471253	.9473116
married						
Married	1.108954	.1213525	0.95	0.345	.8948841	1.374233
kid5	.8579072	.0619658	-2.12	0.034	.7446614	.9883751
phd	.9970707	.0479265	-0.06	0.951	.9074256	1.095572
mentor	1.024022	.0043898	5.54	0.000	1.015454	1.032662
_cons	1.426359	.2807513	1.80	0.071	.969809	2.097836
/lnalpha	-.6034753	.2249915			-1.044451	-.1625001
alpha	.5469076	.1230496			.3518851	.850016

Likelihood-ratio test of $\alpha=0$: $\text{chibar2}(01) = 105.43$ Prob>=chibar2 = 0.000

The output looks like that from `nbreg` except that the title now says "Truncated negative binomial regression" and the truncation point is listed. The LR test provides clear evidence of overdispersion, as was the case with the NBRM. Indeed, the ZTNB is estimating the same parameters as the NBRM but with more limited data. The effects of estimating the parameters with the truncated sample can be seen by comparing the estimates from the two models:

Variable	NBRM	ZTNB
female		
Female	0.805	0.783
	-2.98	-2.52
married		
Married	1.162	1.109
	1.83	0.95
kid5	0.838	0.858
	-3.32	-2.12
phd	1.015	0.997
	0.42	-0.06
mentor	1.030	1.024
	8.38	5.54
_cons	1.292	1.426
	1.85	1.80
lnalpha		
_cons	0.442	0.547
	-6.81	-2.68
Statistics		
N	915	640

legend: b/t

Although the estimates are similar, there is more sampling variability in the ZTNB (indicated by the smaller z -values), which is expected given that estimation uses more limited data.

9.4.2 Interpretation using $E(y | x)$

If counts of zero are missing from your sample because of the way the data were collected, the parameters can be interpreted in the same way as those for the PRM and the NBRM. Essentially, the model fills in the data that were lost when the data were collected. Accordingly, $\exp(\beta_k)$ is the factor increase in the rate for a unit increase in x_k , holding other variables constant. Keep in mind that we are referring to the factor change in the unconditional rate $E(y | x)$ not the conditional rate $E(y | y > 0, x)$.

As with the NBRM and the PRM, you can use the `irr` option (as shown in the output above) or use `listcoef` to compute factor change coefficients after running `tpoisson` or `tnbreg`.


```
. listcoef female mentor
tnbreg (N=640): Factor change in expected count
Observed SD: 1.8823
```

	b	z	P> z	e ^b	e ^b StdX	SDofX
female						
Female	-0.2447	-2.517	0.012	0.783	0.886	0.497
mentor	0.0237	5.538	0.000	1.024	1.278	10.329

Here are some examples of interpreting these results:

Being a female scientist decreases the expected number of papers by a factor of 0.78, holding other variables constant.

Each additional publication by the mentor increases the predicted number of publications by 2.4%, holding other variables constant.

A standard deviation increase in publication by the mentor increases the predicted number of publications by a factor of 1.28, holding other variables constant.⁶

9.4.3 Predictions in the estimation sample

Running `predict` after fitting a truncated count model will generate a new variable with predictions based on the values of the independent variable for each observation. The predictions can be conditional or unconditional. Consider our hypothetical example in which scientists are in the sample only if they have published at least once. Unconditional predictions are predictions about the entire population of scientists, regardless of whether they have published. Conditional predictions are predictions conditional on the count being greater than the truncation point. If we want to make predictions about those who have published (that is, the count is greater than 0), we would use conditional predictions. As noted before, both unconditional and conditional predictions are still conditional on the values of the independent variables.

By default, `predict` computes the unconditional rate. We can obtain the unconditional predicted probability of a specific count $\Pr(y = k)$ with the option `pr(k)` and the predicted probability of a count within a range $\Pr(m \leq y \leq n)$ with the option `pr(m,n)`. We can also compute conditional predictions that are conditional on the count being greater than the truncation point. We obtain conditional predictions about the rate with the option `cm` and conditional predicted probabilities with the option `cpr()` instead of `pr()`.

6. The estimate for the effect of a standard deviation increase in the mentor's productivity is computed as $\exp(\beta_{\text{mentor}} \times \text{sd}_{\text{mentor}})$.

In our example, we use `predict` followed by `summarize` to show the average unconditional and conditional rates and to compare those with the mean number of articles in the sample.

```
. predict rate
(option n assumed; predicted number of events)
. label var rate "Unconditional rate of productivity"
. predict crate, cm
. label var rate "Conditional rate of productivity given at least 1 article"
. summarize rate crate arttrunc
```

Variable	Obs	Mean	Std. Dev.	Min	Max
rate	640	1.687704	.641613	.9468752	8.397242
crate	640	2.425643	.5871667	1.774038	8.774148
arttrunc	640	2.420313	1.882269	1	19

Variable `rate` is the unconditional rate of publication, and `crate` is the conditional rate. The average unconditional rate is smaller because it is estimating the rate for a population that includes scientists with no publications. Because our sample includes only people who have published, the mean of `arttrunc` is very close to the average conditional rate. Recall that to illustrate truncated models, we dropped observations with zero publications. In the full sample, the mean number of articles is 1.693, which is extremely close to the mean of `rate`. Indeed, our model did an excellent job of predicting the unconditional rate.

9.4.4 Interpretation using predicted rates and probabilities

`mtable`, `mgen`, and `mchange` can compute conditional and unconditional rates and predicted probabilities. Unconditional rates are computed by default. Other type of predictions can be made using the `predict()` option. For example, to compute the average conditional rate, you could use the command `mtable, predict(cm)`. To compute unconditional probabilities for counts from m to n , we use the option `predict(pr(m/n))`. To compute conditional probabilities, we use `predict(cpr(m/n))`.

To illustrate this, we compare the conditional and unconditional rates of productivity for married (`married=1`), female (`female=1`) scientists without young children (`kid5=0`), who are average on other characteristics:

```
. qui mtable, rowname(Unconditional) at(female=1 married=1 kid5=0) atmeans ci
. mtable, rowname(Conditional) at(female=1 married=1 kid5=0) atmeans ci
> predict(cm) below brief
```

Expression: Conditional mean of art > ll(1), predict(cm)

	mu	ll	ul
Unconditional	1.561	1.243	1.880
Conditional	2.308	2.062	2.553

An otherwise average married, female scientist without young children is predicted to publish 1.56 papers. However, if we know she has published at least one paper, this pre-

diction increases to 2.31 articles. The conditional rate is higher than the unconditional rate, as it must be, because the conditional rate includes only those with at least one paper. To compute unconditional probabilities from 0 to 5 articles for a married female scientist without children, we type

```
. mtable, at(female=1 married=1 kid5=0) atmeans pr(0/5) ci brief
Expression: Pr(art), predict(pr())
```

	0	1	2	3	4	5
Pr(y)	0.323	0.272	0.177	0.104	0.058	0.031
ll	0.240	0.253	0.150	0.081	0.042	0.020
ul	0.407	0.292	0.205	0.127	0.074	0.042

The probability of a zero count is 0.32 with a 95% confidence interval from 0.24 to 0.41, indicating that we expect that an otherwise average married female scientist with no children would have a 32% chance of having no publications. Other probabilities can be interpreted similarly.

We cannot compute the conditional probability of a zero count because we are conditioning on having at least one publication. Accordingly, we compare unconditional and conditional probabilities of counts greater than 1:

```
. qui mtable, rowname(Unconditional) at(female=1 married=1 kid5=0) atmeans
> pr(1/5)
. mtable, rowname(Conditional) at(female=1 married=1 kid5=0) atmeans
> cpr(1/5) below brief
Expression: Pr(art | art>0), cpr()
```

	1	2	3	4	5
Unconditional	0.272	0.177	0.104	0.058	0.031
Conditional	0.403	0.262	0.154	0.086	0.046

As expected, the conditional probabilities are larger than the unconditional probabilities. This is because the unconditional probability of a 0 is 0.32, so the conditional predictions equal $\Pr(y = k | \mathbf{x}, y > 0) = \Pr(y = k | \mathbf{x}) / \{1 - \Pr(y = 0 | \mathbf{x})\}$. Accordingly, each unconditional probability is 32.3% lower than the corresponding conditional probability.

Although we do not illustrate them, other methods of interpretation that were used for the PRM and NBRM can also be used for count models with truncation.

9.5 (Advanced) The hurdle regression model

This section is marked as advanced because it involves fitting a model that is not built-in to Stata. Obtaining estimates is, consequently, more complicated. The section is useful if you have an application of the hurdle model, and it also provides an opportunity to learn about working with matrices and using the `suest` command to combine results from different models.

The hurdle regression model (HRM) combines a binary model to predict 0s and a ZTP or ZTNB to predict nonzero counts (Mullahy 1986; Cameron and Trivedi [2013, 136–139]). Let y be a count outcome that is not truncated at 0. Suppose that zero counts are generated by a binary process. Here we use a logit to model the binary outcome $y = 0$ versus $y > 0$, but other binary models could be used:

$$\Pr(y_i = 0 \mid \mathbf{x}_i) = \frac{\exp(\mathbf{x}_i \boldsymbol{\gamma})}{1 + \exp(\mathbf{x}_i \boldsymbol{\gamma})} = \pi_i$$

In this two-equation model, 0 is viewed as a hurdle that you have to get past before reaching positive counts. Positive counts are generated either by the ZTP or the ZTNB models from section 9.4. Because there are separate equations to predict zero counts and positive counts, this allows the process that predicts zeros to be different from the process that predicts positive counts.

The name “hurdle” might suggest that the process of moving from 0 to 1 with the count outcome is more difficult than subsequent increases. This makes sense in many count processes, including the case of scientific publishing: we might imagine that getting one’s first publication is especially difficult, and publishing is easier after that. In such cases, the number of zero counts would be greater than we would predict using the PRM or NBRM. Unlike the zero-inflated models considered in section 9.6, however, the hurdle model can also be applied to cases in which there are fewer 0s than expected. In such cases, getting over the hurdle is easier to achieve than subsequent counts.

The predicted rates and probabilities for the HRM are computed by mixing the results of the binary model and the zero-truncated model. The probability of a 0 is

$$\Pr(y_i = 0 \mid \mathbf{x}_i) = \pi_i$$

as estimated by a binary regression model. Because positive counts can occur only if you get past the 0 hurdle, which occurs with probability $1 - \pi_i$, we rescale the conditional probability from the zero-truncated model to compute the unconditional probability

$$\Pr(y_i \mid \mathbf{x}_i) = (1 - \pi_i) \Pr(y_i \mid y_i > 0, \mathbf{x}_i) \quad \text{for } y > 0 \quad (9.10)$$

The unconditional rate is computed by combining the mean rate for those with $y = 0$ (which, of course, is 0) and the mean rate for those with positive counts:

$$E(y_i | \mathbf{x}_i) = (\pi_i \times 0) + \{ (1 - \pi_i) \times E(y_i | y_i > 0, \mathbf{x}_i) \} \quad (9.11)$$

$$= (1 - \pi_i) \times E(y_i | y_i > 0, \mathbf{x}_i) \quad (9.12)$$

where $E(y_i | y_i > 0, \mathbf{x}_i)$ is defined by (9.8) for the ZTP and (9.9) for the ZTNB.

Although Stata does not include the hurdle model as a command, it can be fit by combining results from `logit` with those from either `tnbreg` or `tpoisson`.⁷ Using these estimation commands along with commands for working with predictions, we can compute predictions for the hurdle model that correspond to those for other count models. This process involves a few more steps than the other examples in this book, but these are straightforward and necessary for using this model. The process also provides an example of accomplishing postestimation analyses “by hand”.

A bigger problem is that the standard errors for the parameter estimates and predictions obtained using this method will be incorrect. We will show you how to obtain correct standard errors for the parameters by using Stata’s `suest` command. Unfortunately, these estimates cannot be used with `predict`, `margins`, `mtable`, `mgen`, or `mchange` to obtain predicted probabilities. Accordingly, we show how to make these predictions with estimates from `logit` and `tnbreg`. The predictions will be correct because they do not depend on the standard errors; but the standard errors will be incorrect, so you cannot construct confidence intervals around these predictions or do hypothesis testing.

9.5.1 Fitting the hurdle model

With binary models (see chapter 5), Stata defines the two outcome categories as 0 or any nonmissing value that is not 0. For the hurdle model, this is very convenient. If we use `logit` or `probit` for a count outcome, the resulting estimates are for a binary model of any positive count versus 0, which is precisely what we want for the first step in the hurdle model. Here we fit the logit model and store the estimates:

```
. logit art i.female i.married kid5 phd mentor, nolog or
      (output omitted)
. est store Hlogit
```

Then, we fit a ZTNB by using `if art>0` to restrict our sample to scientists with at least one article:

7. Hilbe (2005) has written a suite of commands that fit a hurdle model. For example, his `hnblogit` command fits a hurdle model that uses a logit model for the 0s and an NBRM for the positive counts. You can find these commands by typing `net search hurdle`. These commands do not work with factor-variable notation.


```
. tnbreg art i.female i.married kid5 phd mentor if art>0, nolog irr
(output omitted)
. est store Hztmb
```

This correctly estimates the coefficients, but the standard errors are incorrect. Accordingly, in the following table, the estimated coefficients are correct but the z -values are not (we only show them for comparison with the correct standard errors, computed with `suest` below):

```
. est table Hlogit Hztmb, b(%9.3f) eform t
```

Variable	Hlogit	Hztmb
art		
female		
Female	0.778	0.783
	-1.58	-2.52
married		
Married	1.386	1.109
	1.80	0.95
kid5		
	0.752	0.858
	-2.57	-2.12
phd		
	1.022	0.997
	0.28	-0.06
mentor		
	1.083	1.024
	6.15	5.54
_cons		
	1.267	1.426
	0.80	1.80
lnalpha		
_cons		0.547
		-2.68

legend: b/t

To obtain the correct standard errors, we use the `suest` command (see [R] `suest`), which takes into account that even though the two models were independently estimated, they are dependent on one another.


```
. suest Hlogit Hztbnb, vce(robust) eform(expB)
```

```
Simultaneous results for Hlogit, Hztbnb
```

Number of obs = 915

	expB	Robust Std. Err.	z	P> z	[95% Conf. Interval]	
Hlogit_art						
female						
Female	.7779048	.1215446	-1.61	0.108	.5727031	1.056631
married						
Married	1.385739	.2475798	1.83	0.068	.9763455	1.966796
kid5	.7518272	.0831787	-2.58	0.010	.6052643	.93388
phd	1.022468	.0822485	0.28	0.782	.8733296	1.197075
mentor	1.083419	.0154716	5.61	0.000	1.063515	1.114171
_cons	1.267183	.3694752	0.81	0.417	.7155706	2.244016
Hztbnb_art						
female						
Female	.7829619	.0724833	-2.64	0.008	.6530404	.9387312
married						
Married	1.108954	.1169726	0.98	0.327	.9018383	1.363636
kid5	.8579072	.0626125	-2.10	0.036	.743562	.9898365
phd	.9970707	.0504934	-0.06	0.954	.9028585	1.101114
mentor	1.024022	.0050724	4.79	0.000	1.014129	1.034012
_cons	1.426359	.27488	1.84	0.065	.9776649	2.080979
Hztbnb_lnal-a						
_cons	.5469076	.1302053	-2.53	0.011	.3429761	.8720957

You can confirm that the parameter estimates are the same as those shown with `estimates` table earlier but that the *z*-values differ.

With the hurdle model, a variable can be significant in one part of the model but not in the other part. For example, women are not significantly different from men “getting over the hurdle”, but they have a significantly lower rate of publication once over the hurdle. For that matter, coefficients for the same variable can be in opposite directions for the two parts of the model, and there is no need for both parts to include the same independent variables.

The logit coefficients can be interpreted as discussed in chapter 6. If the coefficient for x_k is positive, it means that a unit increase in x_k increases the odds of publishing one or more papers by a factor of $\exp(\hat{\beta}_k)$, holding other variables constant. For example, consider the effect that young children have on publishing:

For an additional young child, the odds of publishing at least one article decrease by a factor of 0.75, holding other variables constant.

The ZTNB part of the model estimates how independent variables affect the rate of publication for those who have gotten “over the hurdle” of publishing. The coefficients

from the ZTP or ZTNB do not have a direct substantive interpretation because we are not using the truncated model to overcome limitations of a truncated sample. If there were no 0s in our sample because data were only collected if the event occurred at least once, we could interpret these parameters. Here, however, we assume that the process generating zeros is different from the process generating nonzeros. Recall from our discussion of the zero-truncated model that the exponentiated coefficients estimate effects on the unconditional rate of the outcome. In this case, however, we use the logit equation to predict zero counts, so the exponentiated coefficients from the zero-truncated model no longer correspond directly to changes in the unconditional rate.

9.5.2 Predictions in the sample

Using `predict` with the estimates from our logit model, we can compute the predicted probability of observing a positive count for each observation:

```
. estimates restore Hlogit
(results Hlogit are active now)
. predict Hprobt0
(option pr assumed; Pr(art))
. label var Hprobt0 "Pr(y>0)"
```

If we subtract this probability from 1, we get the predicted probability of a zero count:

```
. gen Hprob0 = 1 - Hprobt0
. label var Hprob0 "Pr(y=0)"
```

Averages can be computed using `summarize`:

```
. summarize Hprob0 Hprobt0
```

Variable	Obs	Mean	Std. Dev.	Min	Max
Hprob0	915	.3005464	.1180336	.0015213	.5612055
Hprobt0	915	.6994536	.1180336	.4387945	.9984787

The average predicted probability of a 0 is 0.3005, which is exactly the same as the proportion of the 0s in the sample, as shown with `tabulate art` earlier. When `logit` is used for the binary portion of the model, the average probability of a 0 will always exactly match the observed proportion of 0s.⁸ Accordingly, in the hurdle model, the average predicted probability of a 0 equals the observed proportion of 0s.

To compute predictions of positive counts, we restore the results from the ZTNB and use `predict` with the `cm` option to generate conditional predictions:

```
. estimates restore Hztbnb
(results Hztbnb are active now)
. predict Hcrate, cm
. label var Hcrate "Conditional rate"
```

8. If `probit` is used for the binary model, the average probability of a 0 is close but not exactly the same as the proportion of 0s in the sample.

The conditional rate is the expected number of publications for those who have made it over the hurdle of publishing at least one article. To compute the unconditional rate, we multiply the conditional rate by the probability of having published at least one article, which we estimated using the logit portion of the model.

```
. gen Hrate = Hcrate * Hprobt0
. label var Hrate "Unconditional rate"
```

We use `summarize` to compare the average conditional and unconditional rates:

```
. sum Hcrate Hrate
```

Variable	Obs	Mean	Std. Dev.	Min	Max
Hcrate	915	2.356661	.52643	1.73671	8.774148
Hrate	915	1.697686	.7016245	.7705519	8.7608

We follow the same idea to compute unconditional probabilities for nonzero counts. Because we want to compute predictions for multiple counts, we use a `forvalues` loop. Within the loop, we first compute the conditional predicted probabilities by using `predict`. Next, we multiply the conditional probabilities by the probability from our logit model of having published at least one article:

```
. forvalues icount = 1/8 {
2.     predict Hcprob`icount', cpr(`icount')
3.     label var Hcprob`icount' "Pr(y=`icount'|y>0)"
4.     gen Hprob`icount' = Hcprob`icount' * Hprobt0
5.     label var Hprob`icount' "Pr(y=`icount')"
```

The loop executes the code within braces eight times, successively substituting the values 1–8 for the macro `icount`. The `predict` command uses the `cpr()` option to compute conditional probabilities. The unconditional probabilities are obtained by multiplying conditional probabilities by the probability of a positive count. We use `summarize` to obtain the average unconditional probabilities:

```
. sum Hprob*
```

Variable	Obs	Mean	Std. Dev.	Min	Max
Hprobt0	915	.6994536	.1180336	.4387945	.9984787
Hprob0	915	.3005464	.1180336	.0015213	.5612055
Hprob1	915	.2772253	.0264024	.0672922	.3398655
Hprob2	915	.1783374	.0225025	.07815	.2233984
Hprob3	915	.1053164	.0246465	.0489526	.1597355
Hprob4	915	.0599405	.022171	.0196098	.1201168
Hprob5	915	.0336258	.0178631	.0075859	.0961457
Hprob6	915	.0188354	.0136543	.002865	.0806172
Hprob7	915	.0106266	.0102113	.0010633	.0676752
Hprob8	915	.0060773	.0075988	.0003894	.0604464

The mean predicted probabilities can be compared with the observed predictions, as illustrated for the PRM and NBRM.

9.5.3 Predictions at user-specified values

We can use `mtable` or `margins` to compute predictions at specific values of the independent variables. You should not compute confidence intervals for these predictions: they will be incorrect because they do not use the correct standard errors from `suest`.

For our example, we make predictions for the ideal type of a married, female scientist (`female=1 married=1`) from an elite PhD program (`phd=4.5`) who studied with a mentor with average productivity. We begin by making predictions from the logit portion of the model:

```
. est restore Hlogit
(results Hlogit are active now)
. mtable, at(female=1 married=1 kid5=0 phd=4.5) atmeans
Expression: Pr(art), predict()
Pr(y)
-----
0.753
```

Specified values of covariates

	female	married	kid5	phd	mentor
Current	1	1	0	4.5	8.77

Recall, that `Pr(y)` is the probability of a 1 in the logit model, which corresponds to having one or more publications. Accordingly, the predicted probability of publishing one or more papers for a scientist with these characteristics is 0.753.

Next, we need the probability of a positive count to compute unconditional predictions with (9.10) and (9.12). Although we could simply type `.753` in our do-file, a better way is to save the result as a local macro. Not only does this prevent typing errors, but our code will still work correctly if we change the model or sample and the predicted probability is no longer 0.753. To do this, we use information in the matrix `r(table)` that is returned by `mtable`. If you type `matlist r(table)`, you will see that the prediction we want is in row 1 and column 1 of the matrix. To store this value in the local macro `prgt0`, we use

```
local prgt0 = e1(r(table),1,1)
```

where the `e1()` function extracts the element in row 1 and column 1 of matrix `r(table)`. After we compute conditional probabilities, we will use the macro when computing unconditional probabilities.

Next, we compute conditional probabilities from the truncated portion of the model:

```
. est restore Hztmb
(results Hztmb are active now)
. mtable, at(female=1 married=1 kid5=0 phd=4.5) atmeans noesample cpr(1/8)
Expression: Pr(art | art>0), cpr()
```

	1	2	3	4	5	6	7	8
	0.413	0.264	0.152	0.083	0.044	0.022	0.011	0.006

Specified values of covariates

	female	married	kid5	phd	mentor
Current	1	1	0	4.5	8.77

The `noesample` option is essential; we explain it briefly here, with further discussion below. The predictions for `mtable` above use estimates from `tnbreg`, which was fit using only observations where `art` is nonzero. By default, `atmeans` would use this sample to compute the mean for `mentor`. We want to compute predictions with the mean for `mentor` based on the entire sample not just the estimation sample. The `noesample` option tells `mtable` (and other commands based on `margins`) to use the entire sample.

The conditional probabilities computed by `mtable` were stored in the matrix `r(table)`. To compute unconditional probabilities, we multiply these conditional probabilities by the probability of a positive count, which we saved in the local macro `prgt0`. This can be done simply with a `matrix` command:

```
. matrix Huncond = `prgt0' * r(table)
. matlist Huncond, format(%8.3f) title(Unconditional probabilities) names(col)
Unconditional probabilities
```

	1	2	3	4	5	6	7	8
	0.311	0.199	0.114	0.062	0.033	0.017	0.009	0.004

These predictions are for the ideal type of a married, female scientist with no children, who studied with an average mentor at an elite graduate program. The predicted probabilities of publishing one article is 0.31, two articles is 0.20, and so on. We could extend these analyses to compare these predictions with those for scientists with other characteristics.

9.5.4 Warning regarding sample specification

The two parts of the hurdle model are fit using different samples. The full sample was used to fit the binary model, while the truncated sample was used to fit the zero-truncated model. When computing predictions at specified values of the independent variables—say, the mean—you want the mean for the full sample, not the smaller, truncated sample. Or, if we want average predictions, we want averages for the full sample. By default, `margins`, `mtable`, `mgen`, and `mchange` use the estimation sample. For example, after fitting the ZTNB, `mtable`, `atmeans` computes predictions by using

the means for cases with nonzero outcomes. What we want, however, are the means for the sample used for the binary portion of the model. To avoid problems, we suggest the following steps.

1. Before fitting the binary portion of the model, use the `drop` command to drop cases with missing data or that you would otherwise drop by using an `if` or `in` condition. This was not necessary in our example above because we wanted to use all the cases to fit the model.
2. Fit the truncated model with an `if` condition to drop cases that are 0 on the outcome (for example, `tnbreg art... if art>0`).
3. When using `margins`, `mtable`, or other `margins`-based commands, use the option `noesample`, which specifies that all cases in memory be used to compute averages and other statistics, rather than using only the estimation sample.

9.6 Zero-inflated count models

The NBRM improves upon the underprediction of 0s in the PRM by increasing the conditional variance without changing the conditional mean. The hurdle model addresses the underprediction of 0s by using two equations: a binary model to predict 0s and a zero-truncated model for the remaining counts. We can think of this as allowing the process that generates the first count to be distinct from the process that generates subsequent counts. Zero-inflated count models, introduced by Mullahy (1986) and Lambert (1992), change the mean structure to allow 0s to be generated by two distinct processes.

Consider our example of scientific productivity. The PRM, NBRM, and HRM assume that every scientist has a nonzero rate of publishing. This implies a chance that any given scientist would have no publications, but it also implies a positive probability for all positive counts. The rates and predicted probabilities differ across individuals according to their characteristics, but all scientists could publish even if that probability is quite small. Substantively, this would be unrealistic if some scientists have no opportunity for publishing. For example, scientists might be employed in an industry where publishing is not allowed, or they might hold jobs that do not involve research. As a result, we would observe more scientists with no publications because zero counts reflect a combination of scientists with no probability of publishing and scientists for whom no publications is the result of a probabilistic process.

Zero-inflated models allow for this possibility. A zero-inflated model assumes that there are two latent (that is, unobserved) groups. An individual in the “always 0” group (group *A*) has outcome 0 with a probability of 1, whereas an individual in the “not always 0” group (group *-A*) might have outcome 0, but there is a nonzero probability that the individual has a positive count. For someone with no publications, we cannot determine whether he or she is in group *A* or group *-A*, but we can model the individual’s probability of being in one of the groups based on observed characteristics. This process is developed in three steps.

Step 1. Model membership into the latent groups A and $\neg A$.

Step 2. Model counts for those in group $\neg A$ who can publish.

Step 3. Compute probabilities for each count as a mixture of the probabilities from the two groups.

We consider each step in turn, followed by an example.

Step 1: Membership in group A

Let $A_i = 1$ if someone is in group A or let $A_i = 0$ otherwise. Group membership is a binary outcome that can be modeled using the logit or probit model from chapters 5 and 6:

$$\Pr(A_i = 1 \mid \mathbf{z}_i) = F(\mathbf{z}_i \gamma) = \psi_i \quad (9.13)$$

This is a binary regression model, where the outcome A_i is unobserved and ψ_i is the probability of being in group A for individual i . The z variables are referred to as inflation variables because they inflate (that is, increase) the number of 0s, as shown below. To illustrate (9.13), assume that two variables affect the probability that an individual is in group A and that we model this with the logit equation

$$\Pr(A_i = 1 \mid z_{i1}, z_{i2}) = \psi_i = \frac{\exp(\gamma_0 + \gamma_1 z_{i1} + \gamma_2 z_{i2})}{1 + \exp(\gamma_0 + \gamma_1 z_{i1} + \gamma_2 z_{i2})}$$

If we had an observed variable indicating group membership, this would be a standard, binary logit model. But because group membership is a latent variable, we do not know whether an individual is in group A or group $\neg A$.

Step 2: Counts for those in group $\neg A$

Among those who are not always 0, the probability of each count, including 0, is determined by either a PRM or an NBRM. In the equations that follow, we are conditioning both on the observed x_k 's and on A being equal to 0. Although the x 's can be the same as the z 's in the first step, they could be different. Defining $\mu_i = \exp(\mathbf{x}_i \beta)$, for the zero-inflated Poisson (ZIP) model, we have

$$\Pr(y_i \mid \mathbf{x}_i, A_i = 0) = \frac{e^{-\mu_i} \mu_i^{y_i}}{y_i!}$$

and for the zero-inflated negative binomial (ZINB) model, we have

$$\Pr(y_i \mid \mathbf{x}_i, A_i = 0) = \frac{\Gamma(y_i + \alpha^{-1})}{y_i! \Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu_i} \right)^{\alpha^{-1}} \left(\frac{\mu_i}{\alpha^{-1} + \mu_i} \right)^{y_i}$$

If we knew which observations were in group $\neg A$, these equations would define the PRM and the NBRM. Here the equations apply only to those observations from group $\neg A$, but we do not have an observed variable indicating group membership.

Step 3: Mixing groups A and -A

Predicted probabilities of observed counts are computed by mixing the probabilities from the binary and count models. The simplest way to understand mixing is with an example. Suppose that retirement status is indicated by $r = 1$ for retired folks and $r = 0$ for those not retired, where

$$\Pr(r = 1) = 0.2$$

$$\Pr(r = 0) = 1 - \Pr(r = 1) = 0.8$$

Let y indicate living in a warm climate, with $y = 1$ for yes and $y = 0$ for no. Suppose that the conditional probabilities are

$$\Pr(y = 1 | r = 1) = 0.5$$

$$\Pr(y = 1 | r = 0) = 0.3$$

so people are more likely to live in a warm climate if they are retired. What is the probability of living in a warm climate for the population as a whole? The answer is a mixture of the probabilities for the two groups weighted by the proportion in each group:

$$\begin{aligned}\Pr(y = 1) &= \{\Pr(r = 1) \times \Pr(y = 1 | r = 1)\} \\ &\quad + \{\Pr(r = 0) \times \Pr(y = 1 | r = 0)\} \\ &= (0.2 \times 0.5) + (0.8 \times 0.3) = 0.34\end{aligned}$$

In other words, the two groups are mixed according to their proportions in the population to determine the overall rate. The same thing is done for the zero-inflated models that mix predictions from groups A and $-A$.

The proportion in each group equals

$$\Pr(A_i = 1) = \psi_i$$

$$\Pr(A_i = 0) = 1 - \psi_i$$

from the binary portion of the model. The probabilities of a 0 within each group are

$$\Pr(y_i = 0 | A_i = 1, \mathbf{x}_i, \mathbf{z}_i) = 1 \text{ by definition of the } A \text{ group}$$

$$\Pr(y_i = 0 | A_i = 0, \mathbf{x}_i, \mathbf{z}_i) = \text{prediction from PRM or NBRM portion of model}$$

The overall probability of a 0 mixes the two types of 0s:

$$\begin{aligned}\Pr(y_i = 0 | \mathbf{x}_i, \mathbf{z}_i) &= (\psi_i \times 1) + \{(1 - \psi_i) \times \Pr(y_i = 0 | \mathbf{x}_i, A_i = 0)\} \\ &= \psi_i + \{(1 - \psi_i) \times \Pr(y_i = 0 | \mathbf{x}_i, A_i = 0)\}\end{aligned}$$

For count outcomes other than 0,

$$\begin{aligned}\Pr(y_i = k | \mathbf{x}_i, \mathbf{z}_i) &= (\psi_i \times 0) + \{(1 - \psi_i) \times \Pr(y_i = k | \mathbf{x}_i, A_i = 0)\} \\ &= (1 - \psi_i) \times \Pr(y_i = k | \mathbf{x}_i, A_i = 0)\end{aligned}$$

where we use the assumption that the probability of a positive count in group A is by definition 0.

Expected counts are computed similarly:

$$\begin{aligned} E(y_i | \mathbf{x}_i, \mathbf{z}_i) &= (0 \times \psi_i) + \{\mu_i \times (1 - \psi_i)\} \\ &= \mu_i (1 - \psi_i) \end{aligned}$$

Because $0 \leq \psi \leq 1$, the expected value must be smaller than μ , which shows that the mean structure in zero-inflated models differs from that in the PRM or NBRM.

The ZIP and ZINB differ in their assumption about the distribution of the count outcome for members of group $-A$. The ZIP assumes a conditional Poisson distribution with a mean indicated by the count equation, while the ZINB assumes a conditional negative binomial distribution that also includes a dispersion parameter that is fit along with the other parameters of the model.

9.6.1 Estimation using `zinb` and `zip`

The ZIP and ZINB models are fit with the `zip` and `zinb` commands, respectively, listed here with their basic options:

```
zip depvar [indepvars] [if] [in] [weight], inflate(indepvars2) [noconstant
    probit vce(vcetype) irr vuong]
```

```
zinb depvar [indepvars] [if] [in] [weight], inflate(indepvars2)
    [noconstant probit vce(vcetype) irr vuong]
```

Variable lists

depvar is the dependent variable, which must be a count with no negative values or nonintegers.

indepvars is a list of independent variables that determine counts among those who are not always 0s. If *indepvars* is not included, a model with only an intercept is fit.

indepvars2 is a list of inflation variables that determine whether one is in the “always 0” group (group A) or the “not always 0” group (group $-A$).

indepvars and *indepvars2* can be the same variables but do not have to be.

Options

Here we consider only those options that differ from the options for earlier models in this chapter.

`probit` specifies that the model determining the probability of being in group *A* versus group *-A* be a binary probit model. By default, a binary logit model is used.

`vuong` requests a Vuong (1989) test of the ZIP versus the PRM or of the ZINB versus the NBRM. Details are given in section 9.7.2. The `vuong` option is not available if robust standard errors are used.

9.6.2 Example of zero-inflated models

The output from `zip` and `zinb` is similar, so here we show only the output for `zinb`:

```
. zinb art i.female i.married kid5 phd mentor,
>   inflate(i.female i.married kid5 phd mentor) nolog
Zero-inflated negative binomial regression      Number of obs   =       915
                                                Nonzero obs      =       640
                                                Zero obs         =       275

Inflation model = logit                      LR chi2(5)        =       67.97
Log likelihood = -1549.991                   Prob > chi2       =       0.0000
```

	art	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
art	female						
	Female	-.1955068	.0755926	-2.59	0.010	-.3436655	-.0473481
	married						
	Married	.0975826	.084452	1.16	0.248	-.0679402	.2631054
	kid5	-.1517325	.054206	-2.80	0.005	-.2579744	-.0454906
	phd	-.0007001	.0362696	-0.02	0.985	-.0717872	.0703869
	mentor	.0247862	.0034924	7.10	0.000	.0179412	.0316312
	_cons	.4167466	.1435962	2.90	0.004	.1353032	.69819
inflate	female						
	Female	.6359328	.8489175	0.75	0.454	-1.027915	2.299781
	married						
	Married	-1.499469	.9386701	-1.60	0.110	-3.339228	.3402909
	kid5	.6284274	.4427825	1.42	0.156	-.2394105	1.496265
	phd	-.0377153	.3080086	-0.12	0.903	-.641401	.5659705
	mentor	-.8822932	.3162276	-2.79	0.005	-1.502088	-.2624984
	_cons	-.1916865	1.322821	-0.14	0.885	-2.784368	2.400995
/lnalpha		-.9763565	.1354679	-7.21	0.000	-1.241869	-.7108443
alpha		.3766811	.0510282			.288844	.4912293

The top set of coefficients, labeled **art** in the left margin, corresponds to the NBRM for those in group *-A*. The lower set of coefficients, labeled **inflate**, corresponds to the binary model predicting group membership.

9.6.3 Interpretation of coefficients

When interpreting zero-inflated models, it is easy to be confused by the direction of the coefficients. `listcoef` makes interpretation simpler. For example, consider the results for the ZINB:

```
. listcoef, help
zinb (N=915): Factor change in expected count
    Observed SD: 1.9261
Count equation: Factor change in expected count for those not always 0
```

	b	z	P> z	e ^b	e ^b StdX	SDofX
female						
Female	-0.1955	-2.586	0.010	0.822	0.907	0.499
married						
Married	0.0976	1.155	0.248	1.103	1.047	0.473
kid5	-0.1517	-2.799	0.005	0.859	0.890	0.765
phd	-0.0007	-0.019	0.985	0.999	0.999	0.984
mentor	0.0248	7.097	0.000	1.025	1.265	9.484
constant	0.4167	2.902	0.004	.	.	.
alpha						
lnalpha	-0.9764
alpha	0.3767

```

    b = raw coefficient
    z = z-score for test of b=0
    P>|z| = p-value for z-test
    eb = exp(b) = factor change in expected count for unit increase in X
    ebStdX = exp(b*SD of X) = change in expected count for SD increase in X
    SDofX = standard deviation of X
Binary equation: factor change in odds of always 0
```

	b	z	P> z	e ^b	e ^b StdX	SDofX
female						
Female	0.6359	0.749	0.454	1.889	1.373	0.499
married						
Married	-1.4995	-1.597	0.110	0.223	0.492	0.473
kid5	0.6284	1.419	0.156	1.875	1.617	0.765
phd	-0.0377	-0.122	0.903	0.963	0.964	0.984
mentor	-0.8823	-2.790	0.005	0.414	0.000	9.484
constant	-0.1917	-0.145	0.885	.	.	.

```

    b = raw coefficient
    z = z-score for test of b=0
    P>|z| = p-value for z-test
    eb = exp(b) = factor change in odds for unit increase in X
    ebStdX = exp(b*SD of X) = change in odds for SD increase in X
    SDofX = standard deviation of X
```

The top half of the output, labeled `Count equation`, contains coefficients for the factor change in the expected count for those who have the opportunity to publish (that is,

group -A). The coefficients can be interpreted in the same way as coefficients from the PRM or the NBRM. For example,

Among those who have the opportunity to publish, being a female scientist decreases the expected rate of publication by a factor of 0.82, holding other variables constant.

The bottom half of the output, labeled **Binary equation**, contains coefficients for the factor change in the odds of being in group A compared with being group -A. These can be interpreted just as the coefficients for a binary logit model. For example,

Being a female scientist increases the odds of not having the opportunity to publish by a factor of 1.89, holding other variables constant.

As found in this example, when the same variables are included in both equations, the signs of the corresponding coefficients from the binary equation are often in the opposite direction of those from the count equation. This makes substantive sense. The count equation predicts number of publications, so a positive coefficient indicates higher productivity. In contrast, the binary equation is predicting membership in the group that always has zero counts, so a positive coefficient implies lower productivity. This is not, however, required by the model.

9.6.4 Interpretation of predicted probabilities

For the ZIP,

$$\widehat{\Pr}(y = 0 \mid \mathbf{x}, \mathbf{z}) = \hat{\psi} + (1 - \hat{\psi}) e^{-\hat{\mu}}$$

where $\hat{\mu} = \exp(\mathbf{x}\hat{\beta})$ and $\hat{\psi} = F(\mathbf{z}\hat{\gamma})$. The predicted probability of a positive count applies only to the $1 - \hat{\psi}$ observations in group -A:

$$\widehat{\Pr}(y \mid \mathbf{x}) = (1 - \hat{\psi}) \frac{e^{-\hat{\mu}} \hat{\mu}^y}{y!}$$

Similarly, for the ZINB,

$$\widehat{\Pr}(y = 0 \mid \mathbf{x}, \mathbf{z}) = \hat{\psi} + (1 - \hat{\psi}) \left(\frac{\hat{\alpha}^{-1}}{\hat{\alpha}^{-1} + \hat{\mu}} \right)^{\hat{\alpha}^{-1}}$$

and the predicted probability for a positive count is

$$\widehat{\Pr}(y \mid \mathbf{x}) = (1 - \hat{\psi}) \frac{\Gamma(y + \hat{\alpha}^{-1})}{y! \Gamma(\hat{\alpha}^{-1})} \left(\frac{\hat{\alpha}^{-1}}{\hat{\alpha}^{-1} + \hat{\mu}} \right)^{\hat{\alpha}^{-1}} \left(\frac{\hat{\mu}}{\hat{\alpha}^{-1} + \hat{\mu}} \right)^y$$

Predicted probabilities can be computed with `margins` and with our `SPost` commands `mtable`, `mgen`, and `mchange`.

Predicted probabilities with mtable

Suppose that we want to compare the predicted probabilities for a married female scientist with young children who came from a weak graduate program with those for a married male from a strong department who had a productive mentor. We can use the `mtable` command to display these two predictions in one table.

First, we use `atvars()` to include in the table the variables on which we are focusing. By default, these would not be included because they do not vary within either of the `mtable` commands. Second, because the `at()` variables in the table make it clear what each row contains, we use `norownumbers` so that only column labels are shown. The `width(7)` option makes the results fit without wrapping.

```
. quietly mtable, at(female=0 married=1 kid5=3 phd=3 mentor=10)
> atvars(female phd mentor) pr(0/5)
. mtable, at(female=1 married=1 kid5=3 phd=1 mentor=0)
> atvars(female phd mentor) pr(0/5) below width(7) norownumbers
Expression: Pr(art), predict(pr())
```

1.	female	phd	mentor	0	1	2	3	4	5
	0	3	10	0.334	0.300	0.185	0.097	0.047	0.021
	1	1	0	0.835	0.096	0.043	0.017	0.006	0.002

Specified values of covariates

	female	married	kid5	phd	mentor
Set 1	0	1	3	3	10
Current	1	1	3	1	0

The predicted probabilities of a 0 include both scientists from group *A* and scientists from group *-A* who by chance did not publish.

To compute the probability of being in group *A*, we use the `predict(pr)` option:⁹

```
. quietly mtable, at(female=0 married=1 kid5=3 phd=3 mentor=10)
> atvars(female phd mentor) predict(pr)
. mtable, at(female=1 married=1 kid5=3 phd=1 mentor=0)
> atvars(female phd mentor) predict(pr) below norownumbers decimals(4)
Expression: Pr(art = always 0), predict(pr)
```

1.	female	phd	mentor	PrAll0
	0	3	10	0.0002
	1	1	0	0.6883

Specified values of covariates

	female	married	kid5	phd	mentor
Set 1	0	1	3	3	10
Current	1	1	3	1	0

9. To determine that this is the option needed to compute the probability of always being 0, we typed `help zinb`, clicked on the blue `zinb` postestimation link, and then clicked on the blue `predict` link.

We used the option `decimals(4)` to show that the probability of being in group *A* for the men is small but is not 0.

Plotting predicted probabilities with `mgen`

In this section, we explore the two sources of 0s and how they each contribute to the predicted proportion of 0s as the number of publications by a scientist's mentor changes. First, we use `mgen` to compute the predicted probability of a 0 of either type as mentor's publications range from 0 to 6, holding other variables to their means:

```
. mgen, at(mentor=(0/6)) atmeans pr(0) stub(ZINB) replace
Predictions from: margins, at(mentor=(0/6)) atmeans predict(pr(0))
```

Variable	Obs	Unique	Mean	Min	Max	Label
ZINBprany0	7	7	.3711116	.2896935	.5536149	prany(y=0) from margins
ZINBll0	7	7	.316322	.2592648	.4426188	95% lower limit
ZINBul0	7	7	.4259011	.3201222	.664611	95% upper limit
ZINBmentor	7	7	3	0	6	Mentor's # of articles
ZINBCprany0	7	7	.3711116	.2896935	.5536149	pr(y<=0)


```
Specified values of covariates
```

1.	1.		
female	married	kid5	phd
.4601093	.6622951	.495082	3.103109

Next, we compute the probability of being in group *A* by specifying the `predict(pr)` option.

```
. mgen, at(mentor=(0/6)) atmeans predict(pr) stub(ZINB) replace
Predictions from: margins, at(mentor=(0/6)) atmeans predict(pr)
```

Variable	Obs	Unique	Mean	Min	Max	Label
ZINBprall0	7	7	.0912045	.0024928	.3322358	Pr(art=always0) from m...
ZINBll	7	7	.0154536	-.0287807	.1578766	95% lower limit
ZINBul	7	7	.1669554	.0116607	.506595	95% upper limit
ZINBmentor	7	7	3	0	6	Mentor's # of articles


```
Specified values of covariates
```

1.	1.		
female	married	kid5	phd
.4601093	.6622951	.495082	3.103109

Variable `ZINBprall0` contains the probability of a 0 from being in group *A*. If we subtract this from the overall probability of a 0 from any source, contained in the variable `ZINBprany0`, we obtain the probability of a 0 for those in group *-A*; these scientists could have published but by chance did not. After we compute this difference, we label the variables so that the legend of our plot will be clear.


```

. gen ZINBprcount0 = ZINBprany0 - ZINBprall0
(908 missing values generated)
. label var ZINBprall0 "Always Zero from Binary Equation"
. label var ZINBprcount0 "Sometimes Zero from Count Equation"
. label var ZINBprany0 "Zeroes from Both Equations"
. label var ZINBmentor "Mentor's Publications"

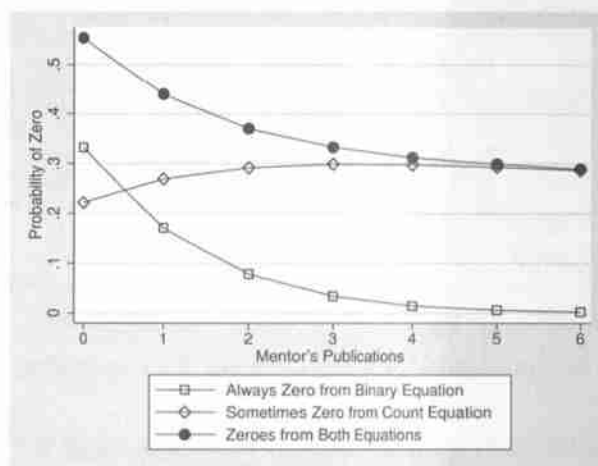
```

These variables can then be plotted:

```

. graph twoway connected ZINBprall0 ZINBprcount0 ZINBprany0 ZINBmentor,
>   xlabel(0/6) ylabel(0(.1).5, gmax)
>   ytitle(Probability of Zero) msymbol(Sh Dh O) legend(rows(3))

```



The curve marked with \square 's is a probability curve like those in chapters 5 and 6 for binary models. It indicates the probability of being in the group that never publishes, where each point corresponds to different rates μ determined by the level of the mentor's publications. The curve marked with \diamond 's shows the probability of 0s from a negative binomial distribution. The overall probability of a zero count is the sum of the two curves, which is shown by the curve with \bullet 's. We see that the probability of being in group A drops off quickly as the mentor's number of publications increases, so for scientists whose mentors have more than three publications, virtually all the zero counts are due to membership in group -A.

9.7 Comparisons among count models

There are two approaches that can be used to compare count models. First, we can compare the mean predicted probabilities and the observed proportions for each count. This was done when we compared predictions from the PRM and NBRM earlier. Second,

we can use various tests and measures of fit to compare models, such as the LR test of overdispersion or the BIC statistic.¹⁰

We begin by showing you how to make these computations using official Stata commands. Then, we demonstrate the `SPost` command `countfit`, which automates this process. Although `countfit` is the simplest way to compare models, it is useful to understand how these computations are made to more fully understand the output of `countfit`.

9.7.1 Comparing mean probabilities

One way to compare count models is to compute average predicted probabilities and compare their fit to the observed data across models. First, we compute the mean predicted probability. For example, in the PRM,

$$\overline{\text{Pr}}_{\text{PRM}}(y = k) = \frac{1}{N} \sum_{i=1}^N \widehat{\text{Pr}}_{\text{PRM}}(y_i = k \mid \mathbf{x}_i)$$

This is simply the average across all observations of the probability of each count. $\widehat{\text{Pr}}_{\text{Observed}}(y = k)$ is the observed probability or proportion of observations with the count equal to k . The difference between the observed probability and the mean probability is

$$\Delta \overline{\text{Pr}}_{\text{PRM}}(y = k) = \widehat{\text{Pr}}_{\text{Observed}}(y = k) - \overline{\text{Pr}}_{\text{PRM}}(y = k)$$

To compute these measures, we first fit each of the four models and then use `mgen`, `meanpred` to create variables containing average predictions:

```
. poisson art i.female i.married kid5 phd mentor, nolog
    (output omitted)
. mgen, stub(PRM) pr(0/9) meanpred
    (output omitted)

. nbreg art i.female i.married kid5 phd mentor, nolog
    (output omitted)
. mgen, stub(NBRM) pr(0/9) meanpred
    (output omitted)

. zip art i.female i.married kid5 phd mentor,
>   inflate(i.female i.married kid5 phd mentor) nolog
    (output omitted)
. mgen, stub(ZIP) pr(0/9) meanpred
    (output omitted)

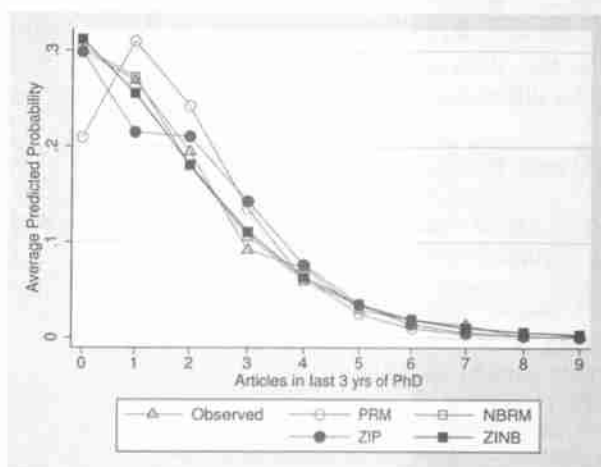
. zinb art i.female i.married kid5 phd mentor,
>   inflate(i.female i.married kid5 phd mentor) nolog
    (output omitted)
```

10. Note that many of these statistics cannot be computed when robust standard errors are used.

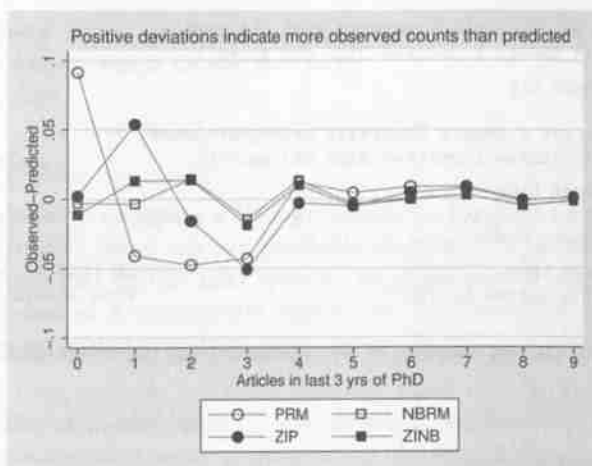

```
. mgen, stub(ZINB) pr(0/9) meanpred
      (output omitted)
```

After each model, `mgen` generates the variable `stubpreq` that contains the average predicted probability $\Pr(y = k)$ for counts 0–9 (specified with the `pr(0/9)` option) and the variable `stubobeq` that contains the observed proportion $\widehat{\Pr}_{\text{Observed}}(y = k)$. The value of the count itself is stored in variable `stubval`. Using these variables, we create a plot that compares the four models with the observed data:

```
. label variable PRMObeq "Observed"
. label variable PRMpreq "PRM"
. label variable NBRMpreq "NBRM"
. label variable ZIPpreq "ZIP"
. label variable ZINBpreq "ZINB"
. graph twoway connected PRMObeq PRMpreq NBRMpreq ZIPpreq ZINBpreq PRMval,
>   ytitle(Average Predicted Probability)
>   xlabel(0/9) msymbol(Th Oh Sh O S) legend(col(3) holes(4))
```



The graph is not very effective because the lines are difficult to distinguish from each other. The information is much clearer if we can instead plot the differences between the observed and predicted probabilities:



Points above 0 on the y axis indicate more observed counts than predicted on average; those below 0 indicate fewer observed counts than predicted.

The graph shows that only the PRM has a problem predicting the average number of 0s. The ZIP does less well, predicting too many 1s and too few 2s and 3s. The NBRM and ZINB do about equally well. From these results, we might prefer the NBRM because it is simpler. Section 9.7.3 shows how to automate the creation of this type of graph with the `countfit` command.

9.7.2 Tests to compare count models

Plotting predictions is only an informal method of assessing the fit of a count model. More formal testing can be done with an LR test of overdispersion and a Vuong test to compare two models.

LR tests of α

Because the NBRM reduces to the PRM when $\alpha = 0$, the PRM and NBRM can be compared by testing $H_0: \alpha = 0$. As shown in section 9.3.3, we find that

```
Likelihood-ratio test of alpha=0:  chibar2(01) = 180.20 Prob>=chibar2 = 0.000
```

which provides strong evidence for preferring the NBRM over the PRM. When robust standard errors are used, estimates are based on pseudolikelihoods rather than likelihoods, and an LR test is not available.

Because the ZIP and ZINB are also nested, the same LR test can be used to compare them. By default, Stata's `lrtest` command will not compare two models that are fit using different estimation commands; however, you can override this with the `force` option. When you do so, the onus is on you to affirm that the estimation samples used are the same and that the LR test is otherwise valid.


```

. quietly zip art i.female i.married kid5 phd mentor,
>   inflate(i.female i.married kid5 phd mentor)
. estimates store zip

. quietly zinb art i.female i.married kid5 phd mentor,
>   inflate(i.female i.married kid5 phd mentor)
. estimates store zinb

. lrtest zip zinb, force
Likelihood-ratio test                LR chi2(1) =    109.56
(Assumption: zip nested in zinb)     Prob > chi2 =    0.0000

```

Given the significant LR test statistics, we conclude that the ZINB significantly improves the fit over the ZIP.

Vuong test of nonnested models

Greene (1994) points out that the PRM and the ZIP are not nested.¹¹ For the ZIP model to reduce to the PRM, ψ must equal 0, but this does not occur when $\gamma = 0$ because $\psi = F(\mathbf{z}0) = 0.5$. Similarly, the NBRM and the ZINB are not nested. Greene proposes using a test by Vuong (1989, 319) for nonnested models. This test considers two models, where $\widehat{\text{Pr}}_1(y_i | \mathbf{x}_i)$ is the predicted probability of observing y_i in the first model and $\widehat{\text{Pr}}_2(y_i | \mathbf{x}_i)$ is the predicted probability for the second model. If there are inflation variables, we are assuming they are part of \mathbf{x} . Define

$$m_i \equiv \ln \left\{ \frac{\widehat{\text{Pr}}_1(y_i | \mathbf{x}_i)}{\widehat{\text{Pr}}_2(y_i | \mathbf{x}_i)} \right\}$$

and let \bar{m} be the mean and s_m be the standard deviation of m_i . The Vuong statistic to test the hypothesis that $E(m) = 0$ is

$$V = \frac{\sqrt{N} \bar{m}}{s_m} \quad (9.14)$$

V has an asymptotic normal distribution. If $V > 1.96$, the first model is favored; if $V < -1.96$, the second model is favored.

For `zip`, the `vuong` option computes the Vuong statistic comparing the ZIP with the PRM; for `zinb`, the `vuong` option compares the ZINB with the NBRM. For example,

```

. zip art i.female i.married kid5 phd mentor,
>   inflate(i.female i.married kid5 phd mentor) vuong nolog
(output omitted)
Vuong test of zip vs. standard Poisson:      z =      4.18  Pr>z = 0.0000

```

The significant, positive value of V supports the ZIP over the PRM. If you use `listcoef`, you get more guidance in interpreting the result:

11. See Allison (2012a) for an alternative view on the nesting of these models. Even if you agree that these models are nested, the Vuong test can be used.


```
. listcoef, help
zip (N=915): Factor change in expected count
(output omitted)
Vuong Test = 4.18 (p=0.000) favoring ZIP over PRM.
(output omitted)
```

Although it is possible to compute a Vuong statistic to compare other pairs of models, such as ZIP and NBRM, these are not available in Stata. This does not mean they cannot be computed, and in the next section, we show how it can be done with the more complicated case of a comparison against the hurdle model. The Vuong test is not appropriate when robust standard errors are used.

Overall, these tests provide evidence that the ZINB fits the data best. However, when fitting a series of models with no theoretical rationale, it is easy to overfit the data, and the ZINB adds many more parameters to the NBRM. Here the most compelling evidence for the ZINB is that it makes substantive sense. Within science, there are some scientists who for structural reasons cannot publish, but for other scientists, the failure to publish in a given period of time is a matter of chance. This is the basis of the zero-inflated models. The NBRM seems preferable to the PRM, because there are probably unobserved sources of heterogeneity that differentiate the scientists. In sum, the ZINB makes substantive sense and fits the data well. At the same time, the simplicity of the NBRM is also compelling.

(Advanced) Computing the Vuong test with a hurdle model

We mark this section as advanced because it involves computing a test that is not built into Stata. The section is only directly useful if you are working with hurdle models. However, computing the test also provides another illustration of working with predicted values postestimation, as well as of the value of mastering how to use loops in Stata.

You might be interested in computing a Vuong test to compare the ZINB to the HRM we fit earlier. This test is not built into Stata, but we can use what we have explained so far to compute it. Because we are computing the test “by hand”, Stata will not stop us from calculating the test statistic even when it is inappropriate; in particular, the Vuong test should not be computed if robust standard errors have been used.

The Vuong test is based on comparing the predicted probabilities of the count values that were actually observed. In other words, for cases in which $y = k$, we compare $\widehat{\Pr}(y = k)$ across models for all observed values of the count outcome. The first step, then, is to generate the predicted probabilities for every observation for all observed counts. In our data, y ranges from 0 to 19. For the ZINB, computing the predicted probabilities of each count is a relatively straightforward application of a **forvalues** loop:


```
. quietly zinb art i.female i.married kid5 phd mentor,
>   inflate(i.female i.married kid5 phd mentor)
. forvalues icount = 0/19 {
2.     predict ZINB`icount', pr(`icount')
3. }
```

For the HRM, this step is more complicated. We first fit the HRM by separately fitting logit and zero-truncated models and store the results:

```
. quietly logit art i.female i.married kid5 phd mentor
. estimates store Hlogit
. quietly tnbbreg art i.female i.married kid5 phd mentor if art > 0
. estimates store Hztbnb
```

We need the logit results to compute the predicted probability of observing a positive count, which we save in the variable `HRMnot0`. Subtracting from 1 gives us the probability of a 0:

```
. estimates restore Hlogit
(results Hlogit are active now)
. predict HRMnot0
(option pr assumed; Pr(art))
. label var HRMnot0 "HRM prob of non-zero count"
. gen HRM0 = 1 - HRMnot0
. label var HRM0 "HRM prob of zero count"
```

To compute the predicted probabilities of counts 1–19, we calculate the conditional probability of each count and then multiply by `HRMnot0` to compute the unconditional probabilities:

```
. estimates restore Hztbnb
(results Hztbnb are active now)
. forvalues icount = 1/19 {
2.     predict HRM`icount', cpr(`icount')
3.     quietly replace HRM`icount' = HRM`icount' * HRMnot0
4.     label var HRM`icount' "HRM unconditional prob(`icount')"
5. }
```

At this point, we have two sets of predicted probabilities: `ZINB0–ZINB19` and `HRM0–HRM19`. For each model, we need to generate a single variable that contains the predicted probability of the count that was actually observed. Every observation has a probability for all possible counts, but only one count is the one that actually occurred for each scientist. We save the probability of the observed count by first creating an empty variable and then using a loop to assign the predicted probability of the observed count to this variable:


```

. gen ZINBprobs = .
(915 missing values generated)
. label var ZINBprobs "ZINB prob of count that was observed"
. gen HRMprobs = .
(915 missing values generated)
. label var HRMprobs "HRM prob of count that was observed"
. forvalues icount = 0/19 {
2.     quietly replace ZINBprobs = ZINB`icount' if art == `icount'
3.     qui replace HRMprobs = HRM`icount' if art == `icount'
4. }

```

We then summarize these variables:

```

. sum ZINBprobs HRMprobs

```

Variable	Obs	Mean	Std. Dev.	Min	Max
ZINBprobs	915	.2346309	.1369695	.0001224	.7454953
HRMprobs	915	.23238	.1221957	.0002254	.5574288

The means are similar, though the average probability of the observed count is slightly higher for the ZINB, meaning that the ZINB has a higher likelihood than the HRM. The Vuong test from (9.14) formalizes this comparison:

```

. gen mZINB_HRM = ln(ZINBprobs/HRMprobs)
. sum mZINB_HRM

```

Variable	Obs	Mean	Std. Dev.	Min	Max
mZINB_HRM	915	.0028478	.1551608	-.7172124	.9502291

```

. display (r(mean)*sqrt(r(N)))/r(sd)
.5551782

```

The resulting V is 0.56, which is less than 1.96. We conclude that we do not have evidence of a significant difference in fit between the ZINB and the HRM.

9.7.3 Using countfit to compare count models

The `countfit` command automates the analyses described in the last two sections for the PRM, NBRM, ZIP, and ZINB. The command can provide a table of estimates, a table of differences between observed and average estimated probabilities, a graph of these differences, and various tests and measures of fit used to compare count models. The syntax is

```

countfit varlist [if] [in] [, inflate(varlist2) noconstant prm nbreg zip
zinb stub(prefix) replace note(string) nograph nodifferences noprtable
noestimates nofit nodash maxcount(#) noisily]

```


Options for specifying the model

varlist is the variable list for the model, beginning with the count outcome variable.

if *exp* and *in range* specify the sample used for fitting the models.

inflate(varlist2) specifies the inflation variables for *zip* and *zinb*.

noconstant specifies that no constant be included in the model.

Options to select the models to fit

By default, PRM, NBRM, ZIP, and ZINB are all fit. If you want only some of these models, specify the models you want:

prm fits the PRM.

nbreg fits the NBRM.

zip fits the ZIP.

zinb fits the ZINB.

Options to label and save results

stub(prefix) can be up to five letters to prefix the variables that are created and to label the models in the output. This name is placed in front of the type of model (for example, *namePRM*). These labels help keep track of results from multiple specifications of models.

replace replaces variables created by *stub()* if they already exist.

note(string) adds a label to the graph.

Options to control what is printed

nograph suppresses the graph of differences from observed counts.

nodifferences suppresses the table of differences from observed counts.

noprtable suppresses the table of predictions for each model.

noestimates suppresses the table of estimated coefficients.

nofit suppresses the table of fit statistics and tests of fit.

nodash suppresses dashed lines between measures of fit.

maxcount(#) specifies the number of counts to evaluate.

`noisily` includes output from Stata estimation commands; without this option, the results are shown only in the `estimates` table output.

To illustrate what `countfit` does, we use `countfit` with our publication example and discuss the output that is generated. `countfit` estimates each of the models, so our command includes the specification of the outcome, the x variables, and the z variables for zero-inflated models. We do not use any of the options that limit the output that is generated:

```
. countfit art i.female i.married kid5 phd mentor, inflate(i.mentor i.female)
```

First, `countfit` presents estimates of the exponentiated parameters for the four models. As we would expect, the direction of coefficients for a given variable is the same for all models.

Variable		PRM	NBRM	ZIP	ZINB
art	female				
	Female	0.799	0.805	0.811	0.822
		-4.11	-2.98	-3.30	-2.59
	married				
	Married	1.168	1.162	1.109	1.103
		2.53	1.83	1.46	1.16
	# of kids < 6	0.831	0.838	0.866	0.859
		-4.61	-3.32	-3.02	-2.80
	PhD prestige	1.013	1.015	0.994	0.999
		0.49	0.42	-0.20	-0.02
	Mentor's # of articles	1.026	1.030	1.018	1.025
		12.73	8.38	7.89	7.10
	Constant	1.356	1.292	1.898	1.517
		2.96	1.85	5.28	2.90
lnalpha	Constant		0.442		0.377
			-6.81		-7.21
inflate	female				
	Female			1.116	1.889
				0.39	0.75
	married				
	Married			0.702	0.223
				-1.11	-1.60
	# of kids < 6			1.242	1.875
				1.10	1.42
	PhD prestige			1.001	0.963
				0.01	-0.12
	Mentor's # of articles			0.874	0.414
				-2.96	-2.79
	Constant			0.562	0.826
				-1.13	-0.14
Statistics	alpha		0.442		
	N	915	915	915	915
	ll	-1651.056	-1560.958	-1604.773	-1549.991
	bic	3343.026	3169.649	3291.373	3188.628
	aic	3314.113	3135.917	3233.546	3125.982

legend: b/t

Next, `countfit` lists the count for which the deviation between the observed and average predicted probability is the greatest. For the PRM, the biggest problem is the prediction of zero counts, with a difference that is much larger than the maximum for the other models. The average difference between observed and predicted is largest for the PRM (0.026) and smallest for the NBRM (0.006) and ZINB (0.008):

Comparison of Mean Observed and Predicted Count

Model	Maximum Difference	At Value	Mean Diff
PRM	0.091	0	0.026
NBRM	-0.015	3	0.006
ZIP	0.054	1	0.015
ZINB	-0.019	3	0.008

This summary information is expanded with detailed comparisons of observed and predicted probabilities for each model.

PRM: Predicted and actual probabilities

Count	Actual	Predicted	Diff	Pearson
0	0.301	0.209	0.091	36.489
1	0.269	0.310	0.041	4.962
2	0.195	0.242	0.048	8.549
3	0.092	0.135	0.043	12.483
4	0.073	0.061	0.012	2.174
5	0.030	0.025	0.005	0.760
6	0.019	0.010	0.009	6.883
7	0.013	0.004	0.009	17.815
8	0.001	0.002	0.001	0.300
9	0.002	0.001	0.001	1.550
Sum	0.993	0.999	0.259	91.964

NBRM: Predicted and actual probabilities

Count	Actual	Predicted	Diff	Pearson
0	0.301	0.304	0.003	0.028
1	0.269	0.272	0.003	0.039
2	0.195	0.180	0.014	1.066
3	0.092	0.106	0.015	1.818
4	0.073	0.060	0.013	2.753
5	0.030	0.033	0.004	0.348
6	0.019	0.018	0.000	0.004
7	0.013	0.010	0.003	0.719
8	0.001	0.006	0.005	3.593
9	0.002	0.004	0.001	0.456
Sum	0.993	0.993	0.062	10.824

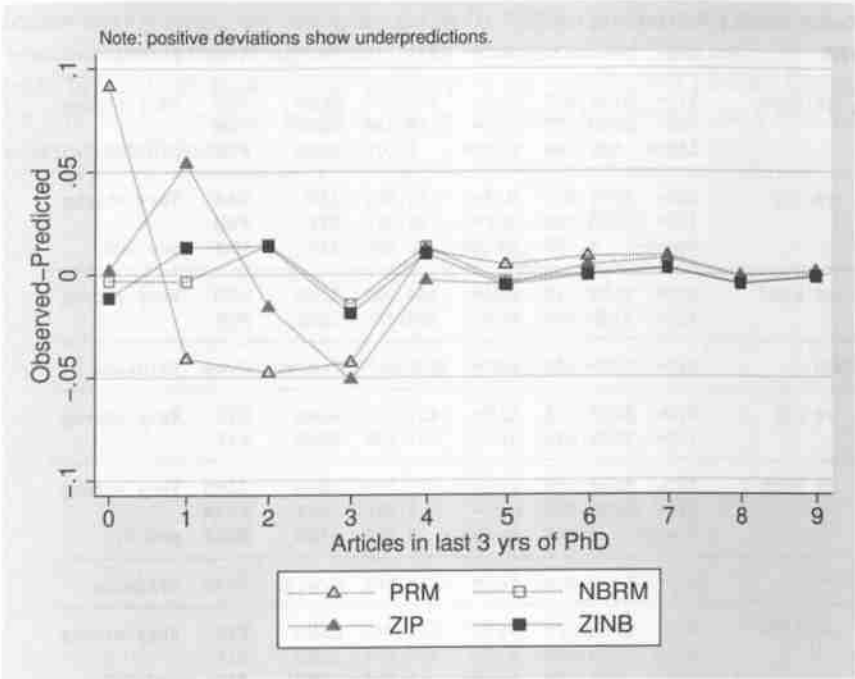
ZIP: Predicted and actual probabilities

Count	Actual	Predicted	Diff	Pearson
0	0.301	0.299	0.002	0.012
1	0.269	0.215	0.054	12.408
2	0.195	0.210	0.016	1.083
3	0.092	0.143	0.051	16.603
4	0.073	0.076	0.003	0.100
5	0.030	0.034	0.005	0.652
6	0.019	0.014	0.005	1.337
7	0.013	0.005	0.008	9.960
8	0.001	0.002	0.001	0.430
9	0.002	0.001	0.001	2.069
Sum	0.993	0.999	0.145	44.653

ZINB: Predicted and actual probabilities

Count	Actual	Predicted	Diff	Pearson
0	0.301	0.312	0.011	0.381
1	0.269	0.256	0.013	0.623
2	0.195	0.181	0.014	0.969
3	0.092	0.111	0.019	2.926
4	0.073	0.063	0.010	1.493
5	0.030	0.035	0.005	0.721
6	0.019	0.019	0.000	0.005
7	0.013	0.010	0.003	0.711
8	0.001	0.006	0.005	3.391
9	0.002	0.003	0.001	0.298
Sum	0.993	0.995	0.081	11.517

The |Diff| columns show the absolute value of the difference between the observed and predicted counts. A plot of these differences is also provided:



Finally, `countfit` compares the fit of the four models by several standard criteria and tests, including AIC and BIC. For each statistic comparing models, the last three columns indicate which model is preferred.

Tests and Fit Statistics						
PRM	BIC= 3343.026	AIC= 3314.113	Prefer	Over	Evidence	
vs NBRM	BIC= 3169.649	dif= 173.377	NBRM	PRM	Very strong p=0.000	
	AIC= 3135.917	dif= 178.196	NBRM	PRM		
	LRX2= 180.196	prob= 0.000	NBRM	PRM		
vs ZIP	BIC= 3291.373	dif= 51.653	ZIP	PRM	Very strong p=0.000	
	AIC= 3233.546	dif= 80.567	ZIP	PRM		
	Vuong= 4.180	prob= 0.000	ZIP	PRM		
vs ZINB	BIC= 3188.628	dif= 154.398	ZINB	PRM	Very strong	
	AIC= 3125.982	dif= 188.131	ZINB	PRM		
NBRM	BIC= 3169.649	AIC= 3135.917	Prefer	Over	Evidence	
vs ZIP	BIC= 3291.373	dif= -121.724	NBRM	ZIP	Very strong	
	AIC= 3233.546	dif= -97.629	NBRM	ZIP		
vs ZINB	BIC= 3188.628	dif= -18.979	NBRM	ZINB	Very strong p=0.012	
	AIC= 3125.982	dif= 9.935	ZINB	NBRM		
	Vuong= 2.242	prob= 0.012	ZINB	NBRM		
ZIP	BIC= 3291.373	AIC= 3233.546	Prefer	Over	Evidence	
vs ZINB	BIC= 3188.628	dif= 102.745	ZINB	ZIP	Very strong p=0.000	
	AIC= 3125.982	dif= 107.564	ZINB	ZIP		
	LRX2= 109.564	prob= 0.000	ZINB	ZIP		

Both the NBRM and the ZINB consistently fit better than either the PRM or the ZIP. This also provides a good example of how BIC penalizes extra parameters more severely than does AIC. The more parsimonious NBRM is preferred over the ZINB according to the BIC statistic but not according to the AIC statistic. The Vuong statistic prefers the ZINB over the NBRM. In terms of fit, there is little to distinguish these two models. If the two-part structure of the ZINB was substantively compelling, we would choose this model. Otherwise, the simplicity of the NBRM would make this the model of choice.

9.8 Conclusion

Count outcomes are not categorical variables in the sense that binary, ordinal, and nominal variables are. Although they are discrete, there is no sense in which the values assigned to a count variable are arbitrary. Indeed, count outcomes support both additive and multiplicative operations, and the number of potential outcome values is not limited.

Even though count variables are thus not categorical, we hope that it is clear how the study of count outcomes can benefit from the same strategies of modeling and interpretation that we introduced in earlier chapters. Instead of only offering interpretations in terms of expected counts, we offered interpretations based on the probabilities of observing a specific count or range of counts. Also, as we showed, the outcomes of 0 can be conceptualized and even modeled as though they are categorically different from the process by which positive counts accumulate. Consequently, as with other types

of outcomes in this book, we can learn much by thinking about and testing our ideas about how outcome values are generated. The combination of Stata with the extra tools we provide in this book make it easy to interpret the relationship between independent variables and count outcomes in ways that are much more effective than vast tables of untransformed coefficients.

References

- Agresti, A. 2010. *Analysis of Ordinal Categorical Data*. 2nd ed. Hoboken, NJ: Wiley.
- . 2013. *Categorical Data Analysis*. 3rd ed. Hoboken, NJ: Wiley.
- Akaike, H. 1973. Information theory and an extension of the maximum likelihood principle. In *Second International Symposium on Information Theory*, ed. B. N. Petrov and F. Csaki, 267–281. Budapest: Akadémiai Kiadó.
- Allison, P. D. 2001. *Missing Data*. Thousand Oaks, CA: Sage.
- . 2012a. Do we really need zero-inflated models? *Statistical Horizons Blog*. <http://www.statisticalhorizons.com/zero-inflated-models>.
- . 2012b. *Logistic Regression Using SAS: Theory and Application*. 2nd ed. Cary, NC: SAS Institute.
- Allison, P. D., and N. A. Christakis. 1994. Logit models for sets of ranked items. *Sociological Methodology* 24: 199–228.
- Amemiya, T. 1981. Qualitative response models: A survey. *Journal of Economic Literature* 19: 1483–1536.
- Anderson, J. A. 1984. Regression and ordered categorical variables (with discussion). *Journal of the Royal Statistical Society, Series B* 46: 1–30.
- Arminger, G. 1995. Specification and estimation of mean structures: regression models. In *Handbook of Statistical Modeling for the Social and Behavioral Sciences*, ed. G. Arminger, C. C. Clogg, and M. E. Sobel, 77–183. New York: Plenum Press.
- Bartus, T. 2005. Estimation of marginal effects using *margeff*. *Stata Journal* 5: 309–329.
- Beggs, S., S. Cardell, and J. A. Hausman. 1981. Assessing the potential demand for electric cars. *Journal of Econometrics* 17: 1–19.
- Brant, R. 1990. Assessing proportionality in the proportional odds model for ordinal logistic regression. *Biometrics* 46: 1171–1178.
- Breen, R., and K. B. Karlson. 2013. Counterfactual causal analysis and nonlinear probability models. In *Handbook of Causal Analysis for Social Research*, ed. S. L. Morgan, 167–187. Dordrecht, Netherlands: Springer.

- Breen, R., K. B. Karlson, and A. Holm. 2013. Total, direct, and indirect effects in logit and probit models. *Sociological Methods and Research* 42: 164–191.
- Buis, M. L. 2007. seqlogit: Stata module to fit a sequential logit model. Boston College Department of Economics, Statistical Software Components S456843. <http://ideas.repec.org/c/boc/bocode/s456843.html>.
- . 2013. oparallel: Stata module providing post-estimation command for testing the parallel regression assumption. Boston College Department of Economics, Statistical Software Components S457720. <http://ideas.repec.org/c/boc/bocode/s457720.html>.
- Bunch, D. S., and R. Kitamura. 1990. Multinomial probit model estimation revisited: Testing of new algorithms and evaluation of alternative model specifications for trinomial models of household car ownership. Research Report UCD-ITS-RP-90-01, Institute of Transportation Studies, University of California, Davis, CA.
- Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.
- . 2010. *Microeconometrics Using Stata*. Rev. ed. College Station, TX: Stata Press.
- . 2013. *Regression Analysis of Count Data*. 2nd ed. New York: Cambridge University Press.
- Cappellari, L., and S. P. Jenkins. 2003. Multivariate probit regression using simulated maximum likelihood. *Stata Journal* 3: 278–294.
- Cattaneo, M. D. 2010. Efficient semiparametric estimation of multi-valued treatment effects under ignorability. *Journal of Econometrics* 155: 138–154.
- Cattaneo, M. D., D. M. Drukker, and A. D. Holland. 2013. Estimation of multivalued treatment effects under conditional independence. *Stata Journal* 13: 407–450.
- Cheng, S., and J. S. Long. 2007. Testing for IIA in the multinomial logit model. *Sociological Methods and Research* 35: 583–600.
- Cleves, M., W. Gould, R. G. Gutierrez, and Y. V. Marchenko. 2010. *An Introduction to Survival Analysis Using Stata, Third Edition*. 3rd ed. College Station, TX: Stata Press.
- Clogg, C. C., and E. S. Shihadeh. 1994. *Statistical Models for Ordinal Variables*. Thousand Oaks, CA: Sage.
- Cook, R. D., and S. Weisberg. 1999. *Applied Regression Including Computing and Graphics*. New York: Wiley.
- Cox, D. R. 1972. Regression models and life-tables (with discussion). *Journal of the Royal Statistical Society, Series B* 34: 187–220.

- Cox, D. R., and E. J. Snell. 1989. *Analysis of Binary Data*. 2nd ed. London: Chapman & Hall.
- Cragg, J. G., and R. S. Uhler. 1970. The demand for automobiles. *Canadian Journal of Econometrics* 3: 386–406.
- Cramer, J. S. 1986. *Econometric Applications of Maximum Likelihood Methods*. Cambridge: Cambridge University Press.
- . 1991. *The Logit Model: An Introduction for Economists*. New York: Chapman and Hall.
- Crow, K. 2013. Export tables to Excel. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2013/09/25/export-tables-to-excel/>.
- . 2014. Retaining an Excel cell's format when using putexcel. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2014/02/04/retaining-an-excel-cells-format-when-using-putexcel/>.
- Cytel Software Corporation. 2005. *LogXact Version 6*. Cambridge, MA.
- Drukker, D. M. 2014. Quantile treatment effect estimation from censored data by regression adjustment. Working paper. <http://www.stata.com/ddrukker/mqgamma.pdf>.
- Efron, B. 1978. Regression and ANOVA with zero-one data: Measures of residual variation. *Journal of the American Statistical Association* 73: 113–121.
- Eliason, S. R. 1993. *Maximum Likelihood Estimation: Logic and Practice*. Newbury Park, CA: Sage.
- Enders, C. K. 2010. *Applied Missing Data Analysis*. New York: Guilford Press.
- Fahrmeir, L., and G. Tutz. 1994. *Multivariate Statistical Modelling Based on Generalized Linear Models*. New York: Springer.
- Fienberg, S. E. 1980. *The Analysis of Cross-Classified Categorical Data*. 2nd ed. Cambridge, MA: MIT Press.
- Firpo, S. 2007. Efficient semiparametric estimation of quantile treatment effects. *Econometrica* 75: 259–276.
- Fox, J. 2008. *Applied Regression Analysis and Generalized Linear Models*. Thousand Oaks, CA: Sage.
- Freedman, D. A. 2006. On the so-called “Huber sandwich estimator” and “robust standard errors”. *American Statistician* 60: 299–302.
- Freese, J. 2002. Least likely observations in regression models for categorical outcomes. *Stata Journal* 2: 296–300.

- Freese, J., and J. S. Long. 2000. sg155: Tests for the multinomial logit model. *Stata Technical Bulletin* 58: 19–25. In *Stata Technical Bulletin Reprints*, vol. 10, 247–255. College Station, TX: Stata Press.
- Fry, T. R. L., and M. N. Harris. 1996. A Monte Carlo study of tests for the independence of irrelevant alternatives property. *Transportation Research Part B: Methodological* 30: 19–30.
- . 1998. Testing for independence of irrelevant alternatives: Some empirical results. *Sociological Methods and Research* 26: 401–423.
- Gallup, J. L. 2012. A programmer's command to build formatted statistical tables. *Stata Journal* 12: 655–673.
- Gould, W., J. Pitblado, and B. Poi. 2010. *Maximum Likelihood Estimation with Stata*. 4th ed. College Station, TX: Stata Press.
- Gould, W. W. 2010. How to successfully ask a question on Statalist. *The Stata Blog: Not Elsewhere Classified*. <http://blog.stata.com/2010/12/14/how-to-successfully-ask-a-question-on-statalist/>.
- Greene, W. H. 1994. Accounting for excess zeros and sample selection in Poisson and negative binomial regression models. Working paper EC-94-10, Department of Economics, Stern School of Business, New York University.
- Greene, W. H., and D. A. Hensher. 2010. *Modeling Ordered Choices: A Primer*. Cambridge: Cambridge University Press.
- Grogger, J. T., and R. T. Carson. 1991. Models for truncated counts. *Journal of Applied Econometrics* 6: 225–238.
- Hagle, T. M., and G. E. Mitchell, II. 1992. Goodness-of-fit measures for probit and logit. *American Journal of Political Science* 36: 762–784.
- Hanmer, M. J., and K. O. Kalkan. 2013. Behind the curve: Clarifying the best approach to calculating predicted probabilities and marginal effects from limited dependent variable models. *American Journal of Political Science* 57: 263–277.
- Hardin, J. W., and J. M. Hilbe. 2012. *Generalized Linear Models and Extensions*. 3rd ed. College Station, TX: Stata Press.
- Hausman, J. A., and D. L. McFadden. 1984. Specification tests for the multinomial logit model. *Econometrica* 52: 1219–1240.
- Heeringa, S. G., B. T. West, and P. A. Berglund. 2010. *Applied Survey Data Analysis*. Boca Raton, FL: Chapman and Hall/CRC.
- Hilbe, J. M. 2005. hnblogit: Stata module to estimate a negative binomial-logit hurdle regression. Boston College Department of Economics, Statistical Software Components S456401. <http://ideas.repec.org/c/boc/bocode/s456401.html>.

- 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.
- Hosmer, D. W., Jr., S. Lemeshow, and R. X. Sturdivant. 2013. *Applied Logistic Regression*. 3rd ed. Hoboken, NJ: Wiley.
- Huebler, F. 2013. Updated programs and guide to integrating Stata and external text editors. *International Education Statistics* (blog). <http://huebler.blogspot.com>.
- Jann, B. 2005. Making regression tables from stored estimates. *Stata Journal* 5: 288–308.
- Karlson, K. B., A. Holm, and R. Breen. 2012. Comparing regression coefficients between same-sample nested models using logit and probit: A new method. *Sociological Methodology* 42: 286–313.
- Kauermann, G., and R. J. Carroll. 2001. A note on the efficiency of sandwich covariance matrix estimation. *Journal of the American Statistical Association* 96: 1387–1396.
- King, G., and M. E. Roberts. 2014. How robust standard errors expose methodological problems they do not fix, and what to do about it. <http://gking.harvard.edu/files/gking/files/robust.pdf>.
- Lall, R., S. J. Walters, and K. Morgan. 2002. A review of ordinal regression models applied on health-related quality of life assessments. *Statistical Methods in Medical Research* 11: 49–67.
- Lambert, D. 1992. Zero-inflated Poisson regression, with an application to defects in manufacturing. *Technometrics* 34: 1–14.
- Landwehr, J. M., D. Pregibon, and A. C. Shoemaker. 1984. Graphical methods for assessing logistic regression models. *Journal of the American Statistical Association* 79: 61–71.
- Lemeshow, S., and D. W. Hosmer, Jr. 1982. A review of goodness of fit statistics for use in the development of logistic regression models. *American Journal of Epidemiology* 115: 92–106.
- Liao, T. F. 1994. *Interpreting Probability Models: Logit, Probit, and Other Generalized Linear Models*. Thousand Oaks, CA: Sage.
- Little, R. J. A., and D. B. Rubin. 2002. *Statistical Analysis with Missing Data*. 2nd ed. New York: Wiley.
- Long, J. S. 1987. A graphical method for the interpretation of multinomial logit analysis. *Sociological Methods and Research* 15: 420–446.
- . 1990. The origins of sex differences in science. *Social Forces* 68: 1297–1316.

- . 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.
- . 2009. *The Workflow of Data Analysis Using Stata*. College Station, TX: Stata Press.
- . Forthcoming. Regression models for nominal and ordinal outcomes. In *Regression Analysis and Causal Inference*, ed. H. Best and C. Wolf. London: Sage.
- Long, J. S., and L. H. Ervin. 2000. Using heteroscedasticity consistent standard errors in the linear regression model. *American Statistician* 54: 217–224.
- Long, J. S., and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*. 2nd ed. College Station, TX: Stata Press.
- Long, J. S., and R. McGinnis. 1981. Organizational context and scientific productivity. *American Sociological Review* 46: 422–442.
- Maddala, G. S. 1983. *Limited-Dependent and Qualitative Variables in Econometrics*. Cambridge: Cambridge University Press.
- McCullagh, P. 1980. Regression models for ordinal data (with discussion). *Journal of the Royal Statistical Society, Series B* 42: 109–142.
- McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. 2nd ed. New York: Chapman and Hall.
- McFadden, D. 1974. Conditional logit analysis of qualitative choice behavior. In *Frontiers of Econometrics*, ed. P. Zarembka, 105–142. New York: Academic Press.
- . 1989. A method of simulated moments for estimation of discrete response models without numerical integration. *Econometrica* 57: 995–1026.
- McKelvey, R. D., and W. Zavoina. 1975. A statistical model for the analysis of ordinal level dependent variables. *Journal of Mathematical Sociology* 4: 103–120.
- Mehta, C. R., and N. R. Patel. 1995. Exact logistic regression: Theory and examples. *Statistics in Medicine* 14: 2143–2160.
- Miller, P. W., and P. A. Volker. 1985. On the determination of occupational attainment and mobility. *Journal of Human Resources* 20: 197–213.
- Mitchell, M. N. 2012a. *Interpreting and Visualizing Regression Models Using Stata*. College Station, TX: Stata Press.
- . 2012b. *A Visual Guide to Stata Graphics*. 3rd ed. College Station, TX: Stata Press.
- Mroz, T. A. 1987. The sensitivity of an empirical model of married women's hours of work to economic and statistical assumptions. *Econometrica* 55: 765–799.

- Mullahy, J. 1986. Specification and testing of some modified count data models. *Journal of Econometrics* 33: 341–365.
- Nagelkerke, N. J. D. 1991. A note on a general definition of the coefficient of determination. *Biometrika* 78: 691–692.
- Peterson, B., and F. E. Harrell, Jr. 1990. Partial proportional odds models for ordinal response variables. *Journal of the Royal Statistical Society, Series C* 39: 205–217.
- Pregibon, D. 1981. Logistic regression diagnostics. *Annals of Statistics* 9: 705–724.
- Punj, G. N., and R. Staelin. 1978. The choice process for graduate business schools. *Journal of Marketing Research* 15: 588–598.
- Rabe-Hesketh, S., and A. Skrondal. 2012. *Multilevel and Longitudinal Modeling Using Stata*. 3rd ed. College Station, TX: Stata Press.
- Raftery, A. E. 1995. Bayesian model selection in social research. In Vol. 25 of *Sociological Methodology*, ed. P. V. Marsden, 111–163. Oxford: Blackwell.
- Schenker, N., and J. F. Gentleman. 2001. On judging the significance of differences by examining the overlap between confidence intervals. *American Statistician* 55: 182–186.
- Sewell, W. H., R. M. Hauser, K. W. Springer, and T. S. Hauser. 2003. As we age: A review of the Wisconsin longitudinal study, 1957–2001. *Research in Social Stratification and Mobility* 20: 3–111.
- Small, K. A., and C. Hsiao. 1985. Multinomial logit specification tests. *International Economic Review* 26: 619–627.
- Sribney, W. M. 1997. FAQ: Why should I not do a likelihood-ratio test after an ML estimation (e.g., logit, probit) with clustering or pweights? <http://www.stata.com/support/faqs/statistics/likelihood-ratio-test/>.
- Stevens, S. S. 1946. On the Theory of Scales of Measurement. *Science* 7: 677–680.
- Theil, H. 1970. On the estimation of relationships involving qualitative variables. *American Journal of Sociology* 76: 103–154.
- Tjur, T. 2009. Coefficients of determination in logistic regression models—A new proposal: The coefficient of discrimination. *American Statistician* 63: 366–372.
- Train, K. 2009. *Discrete Choice Methods with Simulation*. 2nd ed. Cambridge: Cambridge University Press.
- Verlinda, J. A. 2006. A comparison of two common approaches for estimating marginal effects in binary choice models. *Applied Economics Letters* 13: 77–80.
- Vuong, Q. H. 1989. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica* 57: 307–333.

- Weisberg, S. 2005. *Applied Linear Regression*. 3rd ed. New York: Wiley.
- White, H. 1980. A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica* 48: 817–838.
- . 1982. Maximum likelihood estimation of misspecified models. *Econometrica* 50: 1–25.
- Williams, R. 2005. *gologit2*: Stata module to estimate generalized logistic regression models for ordinal dependent variables. Boston College Department of Economics, Statistical Software Components S453401. <http://ideas.repec.org/c/boc/bocode/s453401.html>.
- . 2009. Using heterogeneous choice models to compare logit and probit coefficients across groups. *Sociological Methods and Research* 37: 531–559.
- Windmeijer, F. A. G. 1995. Goodness-of-fit measures in binary choice models. *Econometric Reviews* 14: 101–116.
- Winship, C., and R. D. Mare. 1984. Regression models with ordinal variables. *American Sociological Review* 49: 512–525.
- Winship, C., and L. Radbill. 1994. Sampling weights and regression analysis. *Sociological Methods and Research* 23: 230–257.
- Winter, N. 2000. *smhsiao*: Stata module to conduct Small–Hsiao test for IIA in multinomial logit. Boston College Department of Economics, Statistical Software Components S410701. <http://ideas.repec.org/c/boc/bocode/s410701.html>.
- Wolfe, R. 1998. *sg86*: Continuation-ratio models for ordinal response data. *Stata Technical Bulletin* 44: 18–21. In *Stata Technical Bulletin Reprints*, vol. 8, 149–153. College Station, TX: Stata Press.
- Wolfe, R., and W. Gould. 1998. *sg76*: An approximate likelihood-ratio test for ordinal response models. *Stata Technical Bulletin* 42: 24–27. In *Stata Technical Bulletin Reprints*, vol. 7, 199–204. College Station, TX: Stata Press.
- Wooldridge, J. M. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.
- Xu, J., and J. S. Long. 2005. Confidence intervals for predicted outcomes in regression models for categorical outcomes. *Stata Journal* 5: 537–559.

Author index

A

- Agresti, A. 141, 187, 244, 310
 Akaike, H. 123
 Allison, P. D. 95, 475
 Amemiya, T. 408
 Anderson, J. A. 311, 369, 370, 403,
 445
 Arminger, G. 103

B

- Bartus, T. 245
 Beggs, S. 475
 Berglund, P. A. 100, 101
 Brant, R. 328
 Breen, R. 238, 239
 Buis, M. L. 380
 Bunch, D. S. 475

C

- Cameron, A. C. 9, 21, 84, 85, 104,
 116, 120, 225, 244, 460, 469,
 481, 507, 509, 512, 527
 Cappellari, L. 225
 Cardell, S. 475
 Carroll, R. J. 104
 Carson, R. T. 520
 Cattaneo, M. D. 261
 Cheng, S. 407, 408
 Christakis, N. A. 475
 Cleves, M. 8, 476
 Clogg, C. C. 371
 Cook, R. D. 120, 216
 Cox, D. R. 126, 476
 Cragg, J. G. 127
 Cramer, J. S. 85, 86, 240
 Crow, K. 114
 Cytel Software Corporation. 200

D

- Drukker, D. M. 261

E

- Efron, B. 127
 Eliason, S. R. 84, 85
 Enders, C. K. 95
 Ervin, L. H. 104

F

- Fahrmeir, L. 371
 Fienberg, S. E. 381
 Firpo, S. 261
 Fox, J. 210
 Freedman, D. A. 104
 Freese, J. 9, 217, 398
 Fry, T. R. L. 407, 408

G

- Gallup, J. 113
 Gentleman, J. F. 297
 Gould, W. W. 8, 19, 28, 86, 103, 328,
 476
 Greene, W. H. 245, 328, 371
 Grogger, J. T. 520
 Gutierrez, R. G. 8, 476

H

- Hagle, T. M. 325
 Hanmer, M. J. 244, 245
 Hardin, J. W. 21
 Harrell, F. E., Jr. 374
 Harris, M. N. 407, 408
 Hauser, R. M. 477
 Hauser, T. S. 477
 Hausman, J. A. 407, 409, 475
 Heeringa, S. G. 100, 101

Hensher, D. A. 245, 328, 371
 Hilbe, J. M. 21, 528
 Holland, A. D. 261
 Holm, A. 238, 239
 Hosmer, D. W., Jr. ... 21, 103, 187, 212,
 216, 223, 235, 310

Hsiao, C. 407
 Huebler, F. 38

J

Jann, B. 113
 Jenkins, S. P. 225

K

Kalkan, K. O. 244, 245
 Karlson, K. B. 238, 239
 Kauermann, G. 104
 King, G. 103, 104
 Kitamura, R. 475

L

Lall, R. 374
 Lambert, D. 535
 Landwehr, J. M. 216
 Lemeshow, S. ... 21, 103, 187, 212, 216,
 223, 235, 310
 Liao, T. F. 378
 Little, R. J. A. 95
 Long, J. S. 9, 21, 28, 29,
 41, 84, 85, 104, 123, 187, 188,
 190, 244, 310, 314, 324, 369,
 371, 386, 398, 407, 408, 437,
 460, 467, 481, 483, 507

M

Maddala, G. S. 126, 244
 Marchenko, Y. V. 8, 476
 Mare, R. D. 238, 309
 McCullagh, P. 309, 371
 McFadden, D. L. 106, 126, 407-409,
 469
 McGinnis, R. 437
 McKelvey, R. D. 127, 309
 Mehta, C. R. 199
 Miller, P. W. 309

Mitchell, G. E., II 325
 Mitchell, M. N. 63, 171, 175
 Morgan, K. 374
 Mroz, T. A. 194
 Mullahy, J. 527, 535

N

Nagelkerke, N. J. D. 127
 Nelder, J. A. 371

P

Patel, N. R. 199
 Peterson, B. 374
 Pitblado, J. 86, 103
 Pregibon, D. ... 120, 210, 211, 215, 216
 Punj, G. N. 475

R

Rabe-Hesketh, S. 9, 225
 Radbill, L. 100
 Raftery, A. E. 121, 124, 220
 Roberts, M. E. 103, 104
 Rubin, D. B. 95

S

Schenker, N. 297
 Sewell, W. H. 477
 Shihadeh, E. S. 371
 Shoemaker, A. C. 216
 Skrondal, A. 9, 225
 Small, K. A. 407
 Snell, E. J. 126
 Springer, K. W. 477
 Sribney, W. M. 86, 102, 103
 Staelin, R. 475
 Stevens, S. S. 386
 Sturdivant, R. X. ... 21, 103, 187, 212,
 216, 223, 235, 310

T

Theil, H. 192
 Tjur, T. 127
 Train, K. 22, 188, 469, 475
 Trivedi, P. K. ... 9, 21, 84, 85, 104, 116,
 120, 225, 244, 460, 469, 481,
 507, 509, 512, 527

Tutz, G.....371

U

Uhler, R. S..... 127

V

Verlinda, J. A.....245

Volker, P. A..... 309

Vuong, Q. H..... 539, 548

W

Walters, S. J.....374

Weisberg, S..... 120, 210, 216

West, B. T..... 100, 101

White, H.....103, 105

Williams, R..... 225, 372

Windmeijer, F. A. G.....325

Winship, C.....100, 238, 309

Winter, N.....410

Wolfe, R.....328

Wooldridge, J. M.....22, 244

X

Xu, J..... 244

Z

Zavoina, W..... 127, 309

Subject index

A

- abbreviating commands.....27
- adopath command.....43
- AIC statistic.....see measures of fit,
information criteria
- Akaike information criterion.....
.....see measures of fit,
information criteria
- alternative-specific multinomial probit..
.....469
- AME.....see marginal effects, average
marginal effects
- asclogit command.....457
- ASMNPM.....see alternative-specific
multinomial probit
- asmprobit command.....469
- asobserved option..... see margins
command, options
- assessed categories.....445
- atlegend option..... see margins
command, options
- atmeans option..... see margins
command, options
- atspec specification..... see margins
command, options
- average marginal effects .. see marginal
effects
- aweight modifier.....100

B

- base outcome.... see multinomial logit
model
- baseoutcome() option.....395
- Bayesian information criterion
-see measures of fit,
information criteria

- BIC statistic.....see measures of fit,
information criteria
- binary outcomes.....187
- binary regression models.....187
 - case-control studies.....235
 - comparing logit and probit models
.....196, 207
 - dependent variable coding.....193
 - estimation.....192
 - exact estimation.....85, 199
 - Hosmer-Lemeshow statistic...223
 - hypothesis testing.....200-205
 - identification.....189
 - interpretation, overview.....227
 - latent-variable model.....188
 - logit.....189
 - marginal change in y^*235
 - marginal effects.....239-270
 - measures of fit.....218-224
 - odds ratios.....228
 - other models.....225
 - predictions.....206
 - distribution of.....207
 - graphs.....286-308
 - ideal types.....270-280
 - tables....200-205, 280-284, 286
 - probability.....189, 191, 192
 - probit.....189
 - residuals and influential
observations.....209-218
- biprobbit command.....225
- BLM..... see binary regression models,
logit
- BPM..... see binary regression models,
probit
- brant command.....330
- BRM.....see binary regression models

browse command 25, 46

C

capture command 39

case-control, logit model 235

cd command 29

centile command 251

cformat option 321

clear all command 39, 110

CLM see conditional logit model

cloglog command 225

clonevar command 54

codebook command 50

compact option 47

coefficient of determination see
measures of fit

coefficients

factor change see factor change
coefficients

percentage change .. see percentage
change coefficients

standardized see standardized
coefficients

coeflegend option 90, 117, 202

commands

adopath 43

asclogit 457

asmprobit 469

biprobit 225

brant 330

browse 25, 46

capture 39

cd 29

centile 251

clear all 39, 110

cloglog 225

clonevar 54

codebook 47, 50

constraint 381, 406

countfit 551

datasignature 40

#delimit 36

describe 50

dir 30

display 201

commands, *continued*

do 34

dotplot 49, 207, 340, 413

drop 51

edit 33

egen 54, 95

ereturn list 113, 165

estat classification 128

estat gof 223

estat ic 123, 131

estat mfx 458

estat summarize 109, 130

estat vce 131

estimates 107

estimates describe 111

estimates esample 109

estimates restore 108

estimates save 109

estimates table 111, 196

estimates use 109

exit 39

exlogistic 85, 199

expoisson 85

fitstat 120, 129

foreach 61

forvalues 61, 354, 431

generate 52

global 59

gologit2 372

graph 65

graph combine 361

graph dir 71

graph display 71

graph export 71

graph matrix 78

graph use 71

help 28, 46

hetprobit 225

histogram 263

import sasxport 33

ivprobit 225

keep 51

label 56

label define 57

label value 58

commands, continued

labelbook.....	58
leastlikely.....	217
lincom.....	165, 326
list.....	46
local.....	60
log.....	30
log close.....	31
logit.....	192
lookfor.....	57
lrtest.....	115, 118, 202, 204, 322, 323, 400, 402
ls.....	30
margins.....	139–162, 239, 426
marginsplot...	171, 172, 290, 295
mark.....	94
markout.....	94
matlist.....	166
mchange.....	166
mchangeplot...	346, 417
mgen.....	173
misschk.....	96
misstable.....	96
mllincom.....	165, 275, 354, 427
mllistat.....	148, 288, 426
mlogit.....	390
mlogitplot.....	439, 443
mlogtest.....	398
mprobit.....	465
mtable.....	155–162, 164, 423–431
mvprobit.....	225
nbreg.....	509
net install.....	16
notes.....	41, 50, 59
ocratio.....	381
ologit.....	314
oparallel.....	329
oprobit.....	314
outfile.....	33
poisson.....	488
predict.....	138, 206, 339
probit.....	192
putexcel.....	114
pwcompare.....	231
pwd.....	29

commands, continued

recode.....	55
replace.....	54
reshape.....	456
rologit.....	475
rowmiss.....	95
save.....	32
saveold.....	33
scobit.....	225
search.....	28
seqlogit.....	380, 381
set logtype.....	31
set scheme.....	68
set scrollbufsize.....	26
set seed.....	411
slogit.....	371, 448
spex.....	17
SPost.....	see SPost commands
stcox.....	476
suest.....	529
summarize.....	43, 47
svy.....	100, 107
svyset.....	101
tab1.....	49
tabulate.....	48, 87
test.....	115, 116, 201, 203, 321, 323, 401
testparm.....	117, 204, 323
tnbreg.....	521
tpoisson.....	521
trace.....	102
translate.....	32
twoway.....	63
twoway area.....	363
twoway connected...	67, 175, 361
twoway rarea.....	292
twoway scatter.....	66
update.....	18
use.....	32
verinst.....	18
version.....	39
zinb.....	538
zip.....	538
commands for SPost.....	see SPost commands

- complex survey designs
 - see estimation, complex survey designs
- conditional logit model 454
 - case-specific variables 460
 - data arrangement 455
 - interpretation 458
- conditional predictions
 - zero-truncated models 524
- confidence intervals 103
 - overlapping 297
- constraint** command 381, 406
- continuation ratio model 378–382
- contrasts 231, 252
 - comparing predictions 168
- count outcomes 481
- count regression models 481
 - comparing 544
 - mean predicted probabilities 545
 - measures of fit 551
 - tests 547
 - Vuong test 548
- equidispersion 482, 509
- exact estimation 85
- exposure time 504
- gamma distribution 507
- hurdle regression model 527–535
 - estimation 528
 - predicted probabilities 533
 - predictions 531–535
- negative binomial distribution
 - 508
- negative binomial regression 507–518
 - estimation 509
 - factor change 514
 - marginal effects 515
 - NB1 and NB2 509
 - plotting 518
 - testing overdispersion 511
- observed and predicted counts
 - 484
- observed heterogeneity 507
- count regression models, *continued*
 - overdispersion .. 482, 507, 509, 511
 - truncated models 520
 - percentage change coefficients .. 491
 - Poisson and negative binomial
 - models compared ... 512, 515, 516
 - Poisson distribution 481
 - estimation 483
 - plotting 483, 486
 - Poisson regression 487
 - estimation 488
 - factor change 490
 - incidence-rate ratio ... 489, 491
 - marginal effects 494
 - mgen, meanpred** command 501
 - observed and average predicted probabilities 501
 - observed heterogeneity 487
 - plotting 518
 - plotting probabilities 500
 - predicted probabilities 496
 - rate 490
 - probabilities 516
 - robust standard errors 514
 - testing overdispersion 547
 - truncated counts 518
 - underdispersion 482
 - unobserved heterogeneity 507
 - zero-inflated models 535
 - estimation 538
 - factor change 540
 - latent groups 535
 - negative binomial 538
 - plotting probabilities 543
 - Poisson 538
 - predicted probabilities 541
 - zero-truncated models
 - estimation 521
 - factor change 523
 - negative binomial 520
 - Poisson 519
 - predictions 524–526
- countfit** command 551
- cutpoints 310, 313

D

- data management.....85
 - datasets
 - converting formats.....33
 - entry by hand.....33
 - errors.....34
 - non-Stata formats.....33
 - saving.....32, 79
 - size limitations.....34
 - Stata format.....32
 - using.....32
 - looking at data.....46
 - metadata.....59
 - missing values.....46, 93–98, 193
 - extended.....96
 - numerical ordering.....50
 - selecting observations.....45, 51
 - variable
 - checking transformations.....76
 - creating.....52–59
 - creating dummy.....76, 87
 - creating ordinal.....77
 - describing.....50
 - interaction terms.....89
 - labeling.....56, 76
 - polynomials.....90
 - recoding.....55
 - selecting.....51
 - selection variable.....273
 - summary statistics.....43
 - temporary variables.....88
 - value labels.....57, 92
 - variable lists.....43
 - variable names.....43
 - variable labels.....56
 - workflow principles.....40
- datasets.....see data management; datasets described
- datasets described
 - American National Election Study.....392
 - class identification.....315
 - General Social Survey class identification.....315

datasets described, *continued*

- Health and Retirement Study...101
 - labor force participation.....194
 - mode of travel.....455, 461
 - Mroz.....194
 - party affiliation.....392
 - scientific productivity of
 - biochemists.....483
 - Wisconsin Longitudinal Study....
 -477
 - datasignature** command.....40
 - #delimit** command.....36
 - delta** method.....244
 - describe** command.....50
 - deviance.....see measures of fit
 - dir** command.....30
 - discrete change....see marginal effects
 - dispersion()** option.....509
 - display** command.....201
 - do** command.....34
 - do-files.....34–39
 - comments.....35, 40
 - creating.....37
 - delimiter.....36
 - editor.....37
 - long command lines.....35, 36
 - other editors.....38
 - stopping a do-file.....37
 - syntax highlighting.....37
 - template.....38, 79
 - dotplot** command...49, 207, 340, 413
 - drop** command.....51
 - dydx()** option.....see margins command, options
- E**
- e(sample)** variable.....99
 - edit** command.....33
 - egen** command.....54, 95
 - equidispersion.....see count outcomes
 - ereturn list** command.....113, 165
 - ereturns**.....165
 - error messages.....28, 34
 - errors, **SPost** commands.....18

estat

- classification command....128
- commands130
- gof command.....223
- ic command.....123, 131
- mfx command.....458
- summarize command.....109, 130
- vce command.....131

estimates

- command.....107
- describe command.....111
- esample command.....109
- restore command.....108
- save command.....109
- table command.....111, 196
- use command.....109

estimation

- active estimates.....107
- commands102
 - syntax.....86
 - variable lists87
- complex survey designs....99, 100
- copying results to other programs
 -114

e(sample) variable.....99

estimation sample.....93

postestimation.....98

exact.....85, 199

interaction terms.....89

maximum likelihood84-86

missing values.....93

output105-107

preserving active results.....108

problems.....85

robust standard errors.....103

sample size for ML estimation..85

weights and survey data.....99

exit command.....39

exlogistic command.....85, 199

expoisson command.....85

exposure time.....see count regression
models, exposure time

expressions() option....see margins
command, options

F

factor change coefficients.....179, 184

binary logit model.....228

count regression models

negative binomial regression

model.....515

Poisson regression model....490

zero-inflated regression models..

.....541

zero-truncated regression models

.....524

multinomial logit model.....435

ordered logit model.....335

factor-variable notation....52, 87, 195

test command.....323

allbase option.....196

average marginal effects.....251

base category88

default measurement assumptions

.....89, 163

discrete change.....163

interaction terms.....89, 146

marginal change163

polynomial terms.....90, 146

predictions without.....374

symbolic names90, 202

temporary variables.....88

first difference see marginal effects,
discrete change

fitstat command.....120, 129

binary regression model.....220

ordinal regression model..324-325

foreach command.....61

forvalues command.....61, 354, 431

fweight modifier.....99

G

gamma distribution.....see count
regression

models, gamma distribution

generalized ordered logit model....371

generalized ordered regression model..

.....328

generate command52

mathematical functions.....53

- global command.....59
 - global means.....see predictions
 - GOLM see generalized ordered logit model
 - gologit2 command.....372
 - goodness of fit.....see measures of fit
 - graph
 - combine command.....361
 - command.....65
 - dir command.....71
 - display command.....71
 - export command.....71
 - matrix command.....78
 - use command.....71
 - graph types
 - count distribution.....483, 486
 - cumulative probabilities .. 359, 362
 - histogram.....49
 - index plot.....212
 - influential observation plot...216
 - Lambert plot.....547
 - marginal effects distribution .. 263, 264, 420, 422
 - marginal effects plot.....299, 346, 351, 417
 - observed and average predicted probabilities.....503
 - odds-ratio plot.....436, 437, 439
 - predicted probabilities distribution.....207, 209, 340, 414
 - probabilities ... 172, 175, 286, 287, 289, 292, 359
 - confidence intervals.....292
 - count models.....500, 518, 543
 - local means.....307
 - probabilities for multiple outcomes.....295, 297, 361, 433
 - overlapping CIs.....298
 - residual plot.....212
 - graphs.....63–73
 - area.....363
 - axes.....69
 - captions.....72
 - combining.....72, 361
 - confidence intervals.....292
 - graphs, *continued*
 - connected.....67, 175, 361
 - displaying.....71
 - dotplot.....49, 207
 - exporting.....71
 - file types.....71
 - histogram.....263
 - labeling data points.....212
 - marginsplot.....172
 - matrix.....78
 - menu.....25, 63
 - mgen command.....173
 - multiple predictions.....293–297
 - odds ratios.....436–444
 - overlaying multiple plots.....293
 - overview.....63–73
 - plot options.....67
 - predictions.....171
 - printing.....72
 - probability.....432
 - rarea.....292
 - saving.....70
 - scatterplot.....66
 - schemes.....68
 - titles.....68
 - twoway.....63
 - grouped continuous regression model...445
 - grouped continuous variable.....311
- ## H
- help
 - contacting the authors.....19
 - getting help.....18
 - help command.....28, 46
 - hetprobit command.....225
 - histogram command.....263
 - HL statistic.....
 - see binary regression models, Hosmer–Lemeshow statistic
 - Hosmer–Lemeshow statistic.....see binary regression models, Hosmer–Lemeshow statistic
 - HRM.....see count regression models, hurdle regression model

HTML output 24, 113
 hurdle regression model see
 count regression models, hur-
 dle regression model
 hypothesis testing see testing

I

ideal types *also see* marginal
 effects, ideal types, *see* predic-
 tions, ideal types
 if qualifier 45, 46, 51, 193, 254
 IIA *see* multinomial logit model
 import sasxport command 33
 in qualifier 45, 51, 193
 incidence-rate ratio *see* count
 regression models
 independence of irrelevant
 alternatives .. *see* multinomial
 logit model
 indistinguishable outcomes *see*
 multinomial logit model
 influential observations 209
 Cook's distance 215
 delta-beta influence statistic... 215
 information criteria *see* measures of
 fit
 installing SPost *see* SPost13,
 installing
 interaction terms 89, 146, 241
 marginal effects 259
 ivprobit command 225
 iweight modifier 100

K

keep command 51

L

label
 command 56
 define command 57
 value command 58
 labelbook command 58
 latent-variable model 188, 310
 L^AT_EX output 113
 least likely observations 216

leastlikely command 217
 level() option 102
 likelihood-ratio test *see* testing,
 likelihood-ratio test
 lincom command 165, 326
 linear compared with nonlinear models
 133
 linear probability model 192
 linear regression model 133
 list command 46
 listcoef command 178–184
 binary regression model 228
 count regression models
 negative binomial regression....
 514
 Poisson regression model.... 492
 zero-inflated models 540
 zero-truncated models 523
 multinomial logit model .. 395, 435
 ordinal regression model .. 334, 336
 standardized coefficients 182
 stereotype logistic regression model
 451
 local command 60
 local means *see* predictions
 log close command 31
 log command 30
 log files 30–32, 40, *also see* do-files
 logit command 192
 logit defined 192
 logit model
 binary *see* binary regression
 models
 multinomial *see* multinomial
 logit model
 ordered *see* ordinal regression
 models
 rank-ordered *see* rank-ordered
 logit model
 stereotype .. *see* stereotype logistic
 regression model
 lookfor command 57
 loops 61, 354, 431
 over all observations 420

LR test....see testing, likelihood-ratio test
 LRM.....see linear regression model
 lrtest command...115, 118, 202, 204, 322, 323, 400, 402
 combining alternatives.....406
 ls command.....30

M

m* commands.....see
 mchange command; mgen
 command; mlincom command;
 mlistat command; mtable
 command; SPost13
 compared with margins command
 140

macros

 global.....59
 local.....59
 marginal change...see marginal effects
 marginal effects
 added to odds-ratio plot.....443
 average marginal effects...144, 166, 243, 341, 416
 continuous variables.....248
 factor variables.....251
 interpretation.....248, 252
 binary regression model..239–270
 choosing which measure to use....
 244
 comparing marginal and discrete
 changes.....241
 contrasts.....168, 252
 count regression models
 negative binomial.....515
 Poisson regression.....493
 discrete change.....168
 distribution.....261–270, 420
 general procedures.....265
 ideal types.....278
 tests to compare.....354
 interactions.....241
 interactions and polynomials..259
 linked variables.....241, 259
 marginal change.....168

marginal effects, continued

 marginal effects at representative
 values.....243, 255
 marginal effects at the mean..142,
 166, 243, 255, 341
 margins command.....163
 mchange command.....166
 mtable command.....164
 multinomial logit model..415–423
 ordinal regression model..341–351
 overview.....133, 135, 137, 162
 plotting.....299
 mchangeplot command.....417
 polynomials.....241
 second differences.....285, 426
 standard errors.....244
 subgroups.....254
 summary measures.....242
 summary table.....252
 testing equality of AMEs.....285
 marginal effects at representative values
 see marginal effects
 marginal effects at the mean.....see
 marginal effects
 margins command.....139–162, 239
 compared with m* commands..138,
 140
 if qualifier.....152
 in qualifier.....152
 multiple predictions.....146
 options
 (atstat) suboption.....143
 asobserved.....144, 248
 at().....142, 147, 150, 152
 atlegend.....142
 atmeans.....142
 dydx().....163
 expression().....154
 gen().....267
 noatlegend.....148
 outcome().....159
 over().....151, 152, 283, 430
 post.....165, 426
 predict()...153, 159, 352, 426

- margins command, options, *continued*
 - pwcompare 168, 277
 - varlist 151
 - order of predictions 150
- marginsplot command .. 171, 290, 295
- mark command 94
- markout command 94
- matlist command 166
- maximum likelihood see estimation
- mchange command 166,
 - also see marginal effects
 - binary regression model .. 246–248
 - centered versus uncentered 168
 - count regression models
 - negative binomial regression
 - model 515
 - Poisson regression model... 495, 497
 - default marginal effects... 167, 170
 - interactions 171
 - multinomial logit model 416
 - options 168
 - amount() 168
 - delta() 169
 - statistics() 169
 - trim() 168
 - uncentered 168
 - ordinal regression model 341
 - return matrix 262, 279
- mchangeplot command 346, 417
- measures of fit 120–130
 - binary regression model 218
 - count regression models 551
 - deviance 123
 - formula 123
 - information criteria .. 123, 124, 131
 - difference in BICs 124
 - LR chi-square test of all coefficients 123
 - multinomial logit model 411
 - ordinal regression model .. 324–325
 - pseudo- R^2 126–128
 - ordinal regression model 325
 - R^2 126
- MEM see marginal effects, marginal effects at the mean
- MER see marginal effects, marginal effects at representative values
- mgen command 173
 - binary regression model .. 286–308
 - count regression models
 - negative binomial regression
 - model 517
 - Poisson regression model... 500
 - zero-inflated models 543
 - default predictions 176
 - multinomial logit model 432
 - naming generated variables... 174
 - options 178
 - meanpred 177, 485
 - predlabel() 291
 - replace 174
 - stub() 174, 291
 - ordinal regression model 359
- minimal set 387
- misschk command 96
- missing data see data management, missing data
- missing values
 - estimation commands 93, 98
- misstable command 96
- mllincom command 165, 275, 354
 - second differences 427
- mllistat command 148, 288, 426
- mlogit command 390
- mlogitplot command 439
 - adding marginal effects 443
- mlogtest command 398
 - combining alternatives ... 403, 405
 - independence of irrelevant alternatives 408
 - testing independent variables .. 399
- MNLM see multinomial logit model
- MNPM ... see multinomial probit model
- mprobit command 465
- mtable command 155–162, 355–359
 - binary regression model .. 270–284
 - categorical outcomes 158

mtable command, *continued*

- combining and formatting tables... 160
- conditional and unconditional
 - probabilities... 525
- count outcomes... 158
- count regression models... 497
 - hurdle regression model... 533
- negative binomial regression
 - model... 515
- Poisson regression model... 497–499
- zero-inflated models... 541
- zero-truncated models model... 525
- labeling predictions... 157
- marginal effects... 164
- multinomial logit model... 423–431
- options
 - atright**... 353
 - atvars()**... 158, 162
 - below**... 164
 - brief**... 161
 - clear**... 270
 - colegnm()**... 160
 - decimal()**... 159
 - estname()**... 157
 - long**... 498
 - noesample**... 534
 - norownum**... 162, 353
 - norownumbers**... 498
 - over()**... 357
 - right**... 161
 - rowname()**... 164
 - statistics()**
 - title()**... 162
 - width()**... 353
- ordinal regression model... 351
- stored results... 275
- multinomial logit model... 386
 - asclogit** command for estimation... 462
 - base outcome... 390, 391, 395
 - plotting... 438
 - combining alternatives... 403

multinomial logit model, *continued*

- compared with binary logit model... 389
- compared with ordinal regression
 - model... 413, 433
- estimation... 390
- formal statement of model... 390
- hypothesis testing... 398–406
- independence of irrelevant
 - alternatives... 407–411, 477
 - cautions... 408
 - Hausman–McFadden test... 407, 408
 - Small–Hsiao test... 407, 409
- indistinguishable outcomes... 403
- interpretation, overview... 411
- introduction... 386
- marginal effects... 415–423
 - distribution of... 420
 - interpretation... 418
- measures of fit... 411
- minimal set... 387
- odds ratios... 435
 - interpretation... 435, 441
- odds-ratio plot... 436–444
 - marginal effects... 443
- plotting predictions... 432
- predicted probabilities... 411–415
- predictions
 - distribution of... 414
 - graphs... 417
 - table... 423
- probabilities... 390
- specification searches... 398
- testing independent variables... 399
- multinomial probit model... 465
 - identification... 467
 - with IIA... 465
- mvprobit** command... 225

N

- NB1**... see count regression models, negative binomial regression
- NB2**... see count regression models, negative binomial regression

nbreg command 509
NBRM see count regression models,
 negative binomial regression
 negative binomial distribution see
 count regression
 models, negative binomial
 distribution
net install command 16
noconstant option 102
nofvlabel option 92
nolog option 102
 nominal outcomes 385
 nominal regression models 385
 alternative-specific multinomial
 probit 469
 conditional logit model 454
 multinomial logit model see
 multinomial logit model
 multinomial probit model
 with IIA 465
 stereotype logistic regression model
 445
 nonlinear compared with linear models
 133
 nonlinear models, overview 135
notes command 41, 50, 59
numlist 147

O

observations completely determined ...
 see perfect prediction
ocratio command 381
 odds ratios 179
 binary logit model 228
 compared with marginal effects ...
 444
 confidence intervals 231
 interactions 336
 interactions and polynomials .. 229
 interpretation 229–235
 limitations 234
 multinomial logit model 435
 multiplicative coefficients 233
 ordered logit model 335
 plotting 436–444

odds ratios, *continued*
 reversed ordering 338
 stereotype logistic regression model
 450
 odds-ratio plot see **mlogitplot**
 command
OLM see **ologit** command; ordinal
 regression models
ologit command 314
oparallel command 329
OPM ... see **oprobit** command; ordinal
 regression models
oprobit command 314
 ordered logit model see ordinal
 regression models
 ordered probit model see ordinal
 regression models
 ordered regression models .. see ordinal
 regression models
 ordinal outcomes 309
 ordinal regression models 309
 compared with multinomial logit
 model 413, 433
 continuation ratio model .. 378–382
 criteria for ordinal model 369
 cutpoints 310, 313
 estimation 314
 generalized ordered logit model ...
 371
 hypothesis testing 320–324
 identification 325
 interpretation, overview 331
 latent-variable model 310
 less common models 370
 logit and probit comparison ... 318
 marginal change in y^* 332
 marginal effects 341–351, 364
 summarizing 350
 measures of fit 324–325
 nonlinear probability model ... 314
 odds ratios 335
 ordered logit model 312
 ordered probit model 312

ordinal regression models, *continued*
 parallel regression assumption 326
 caveats 331
 predictions
 distribution of 340
 graphs 344, 359–364
 tables 355–359
 rank-ordered logit model 475
 sequential logit model 378–382
 signs of effects 366
 standardized coefficients 332
 stereotype logistic regression model
 370
 transformed coefficients 332
 ORM see ordinal regression models
 outfile command 33
 outliers 209, see residuals
 over() option see margins
 command, options
 overdispersion see count regression
 models

P

panel data 9
 parallel regression assumption see
 ordinal regression models
 percentage change coefficients 179,
 184
 perfect prediction ... 197, 319, 320, 397
 pformat option 321
 plotting see graphs
 poisson command 488
 Poisson distribution see count
 regression models
 polynomial terms 90, 146, 241
 marginal effects 259
 postestimation commands
 estimation sample 98
 posting estimates see predictions,
 posting
 posting predictions see predictions,
 posting
 pr* commands 12

predict command 138, 206
 binary regression model 206
 count regression models
 hurdle regression model 531
 Poisson regression model 501
 zero-truncated models 524
 default predictions 139
 multinomial logit model 412
 ordinal regression model 339
 predict() option see margins
 command, options
 predicted probabilities see
 mgen command; mtable
 command; mchange command;
 predictions
 predictions 137
 at specified values 139–162
 average prediction 177
 by group 150
 distribution of predictions ... 207,
 340, 414
 each observation 138
 global means 273, 429
 ideal types 270–280, 282, 351
 comparing with tests ... 274, 275
 local means 273, 283
 local means 273, 282, 283, 357,
 429
 graphing 303
 over() option 430
 multiple predictions 146
 nondefault 153
 observed and average predicted
 probabilities 501
 plotting ... 171, 173, 286–308, 344,
 359, 432, 500, 543
 interaction and power terms
 301
 multiple outcomes 175
 multiple predictions 175,
 293–297
 posting 165
 subsamples 429
 tables 155, 280–284, 355, 423

predictions, *continued*
 testing 165
 using margins 141
 predicts perfectly see perfect prediction
 primary sampling unit 100
 PRM see count regression models, Poisson regression
 probability model 192, 314
 probit command 192
 probit model
 binary see binary regression models
 nominal see multinomial probit model
 ordered see ordinal regression models
 profile.do file 27
 proportional-odds assumption see ordinal regression models
 proportional-odds model 309
 pseudo- R^2 see measures of fit
 putexcel command 114
 pwcompare command 231
 pwd command 29
 pweight modifier 99

Q

quasiseperation... see perfect prediction

R

r-returns 202
 R^2 see measures of fit
 count 127
 Cox-Snell 126
 Cragg and Uhler's 127
 Efron's 127
 maximum likelihood 126
 McFadden's 126
 McKelvey and Zavoina's 127
 Nagelkerke 127
 Tjur's coefficient of discrimination 127
 random numbers 411
 random utility model 465

rank-ordered logit model 475
 ties 476
 recode command 55
 reference category see multinomial logit model, base outcome
 regression models, overview 10
 relative-risk ratios 391, 435
 replace command 54
 replication 40
 research diary 40
 reshape command 456
 residuals 209
 robust standard errors 103, 512
 ROLM see rank-ordered logit model
 rologit command 475
 rowmiss command 95

S

save command 32
 saveold command 33
 scalar measures of fit .. see measures of fit
 scobit command 225
 search command 28
 second differences... see marginal effects
 selecting observations see data management
 selecting variables see data management, variable
 seqlogit command 380, 381
 sequential logit model 378-382
 set logtype command 31
 set scheme command 68
 set scrollbufsize command 26
 set seed command 411
 sformat option 321
 SLM .. see stereotype logistic regression model
 slogit command 371, 448
 SMCL... see Stata Markup and Control Language
 SORM see stereotype logistic model
 specification searches 398
 spex command 17

SPost

- citing.....xxii
- support.....18

SPost commands

- brant.....330
- countfit.....551
- fitstat.....120
- leastlikely.....217
- listcoef.....179, 336, 492, 514
- mchange...166, 246, 341, 416, 497
- mchangeplot.....346, 417
- mgen.....173, 359, 485, 500, 543
- mllincom.....165, 275, 354, 427
- mlistat.....148, 288, 426
- mlogitplot.....439
- mlogtest.....398, 403, 408
- mtable.....497, 533, 541

SPost13

- spost13.ado package.....11, 14
 - installing.....13, 14
 - uninstalling.....17
- spost13.do package.....17, 36
 - installing.....17

SPost9.....11

- spost9.ado package.....11
 - uninstalling.....14
- spost9_legacy package.....11

standardized coefficients..179, 236, 332

- fully standardized.....182
- odds ratios.....435
- x*-standardized.....180
- y*-standardized, *y**-standardized ..
.....181

Stata

- abbreviating commands.....27
- ado path.....43
- command syntax.....41-46
- file types.....30
- forum.....29
- getting help.....27-29
- interface.....23
- introduction.....23, 73-80
- Markup and Control Language...
.....31, 32
- NetCourses.....29

Stata, continued

- PDF documentation.....28
- short courses.....29
- Statalist.....29
- tutorial.....73-80
- updating.....12
- versions.....33, 39
- windows.....23-25
- working directory.....29

Stat/Transfer.....33

stcox command.....476

stereotype logistic regression model ...

.....370, 445

- higher dimensional.....453
- identification.....447
- interpretation.....452
- odds ratios.....450
- ordinality.....452

strata.....100

suest command.....529

summarize command.....43, 47

survival analysis.....8

svy command.....100

- vce(cluster *clustvar*) option....
.....100

svy estimation

- output.....107

svyset command.....101

symbolic names.....see factor-variable
notation

syntax

- brant.....330
- countfit.....551
- estimates store.....108
- estimates table.....112
- estimation commands.....86
- fitstat.....121
- leastlikely.....217
- listcoef.....179
- log.....30
- logit.....192
- lrtest.....118
- mlogit.....390
- mlogtest.....398
- nbreg.....509

syntax, *continued*

ologit	314
oprobit	314
poisson	488
predict	138, 206, 339
probit	192
slogit	448
Stata	41–46
tnbreg	521
tpoisson	521
zinb	538
zip	538

T

tab1 command	49
tables	160
tables of estimates	111
tabulate command	48, 87
test command	115, 116, 201, 203, 321, 401
accumulate option	118
combining alternatives	404
factor-variable notation	323
test types	
combining categories	403
Hosmer–Lemeshow statistic	223
ideal types	274, 275, 354
independence of irrelevant alternatives	407
marginal effects	169, 244, 348
multiple coefficients	203, 322
overdispersion	511, 547
parallel regression assumption	328
predictions	166
second differences	285, 426
single coefficients	200, 321
variables	
all	123
multinomial logit model	399
Vuong test	548
testing	114
binary regression model	200–205
independence of irrelevant alternatives	408

testing, *continued*

likelihood-ratio	116, 118, 202, 204
invalid test	120
multinomial logit model	398–406
one-tailed versus two-tailed	115
ordinal regression models	320–324
Vuong test	548
Wald and likelihood-ratio tests	
compared	205
Wald test	115
testparm command	117, 204, 323
thresholds	<i>see</i> cutpoints
tnbreg command	521
tpoisson command	521
trace command	102
translate command	32
truncated count model	<i>see</i> count regression models, zero-truncated models
tutorial	73–80
twoway area command	363
twoway command	63
overlaying multiple plots	293
twoway connected command	361
twoway rarea command	292
twoway scatter command	66

U

underdispersion	<i>see</i> count regression models
update command	18
use command	32

V

value labels	<i>see</i> data management, variable
variable labels	<i>see</i> data management, variable
variables	<i>see</i> data management
vce() option	100, 103
verinst command	18
version command	39

vsquish option.....103
Vuong test.....548

W

Wald test.....see testing, Wald test
weights.....99, 100
workflow for data analysis.....40–41
working directory.....29

Z

zero-inflated negative binomial
 model....see count regression
 models, zero-inflated models
zero-inflated Poisson model.....see
 count regression models, zero-
 inflated models
zero-truncated negative binomial
 model....see count regression
 models, zero-truncated models
zero-truncated Poisson model.....see
 count regression models, zero-
 truncated models
ZINB.....see count regression models,
 zero-inflated models
zinb command.....538
ZIP.....see count regression models,
 zero-inflated models
zip command.....538
ZTNB.....see count regression models,
 zero-truncated models
ZTP.....see count regression models,
 zero-truncated models

The goal of *Regression Models for Categorical Dependent Variables Using Stata, Third Edition* is to make it easier to carry out the computations necessary to fully interpret regression models for categorical outcomes by using Stata's `margins` command. Because the models are nonlinear, they are more complex to interpret. Most software packages that fit these models do not provide options that make it simple to compute the quantities useful for interpretation. In this book, the authors briefly describe the statistical issues involved in interpretation, and then they show how you can use Stata to perform these computations.

Scott Long is Distinguished Professor of Sociology and Statistics at Indiana University—Bloomington. He has contributed articles to many journals, including the *American Sociological Review*, *American Journal of Sociology*, *American Statistician*, and *Sociological Methods and Research*. Dr. Long has authored or edited seven previous books on statistics, including the highly acclaimed *Regression Models for Categorical and Limited Dependent Variables* and *The Workflow of Data Analysis Using Stata*. In 2001, he received the Paul Lazarsfeld Memorial Award for Distinguished Contributions to Sociological Methodology. He is an elected Fellow of the American Statistical Association and a member of the Sociological Research Association.

Jeremy Freese is Ethel and John Lindgren Professor of Sociology and Faculty Fellow of the Institute for Policy Research at Northwestern University. He has previously been a faculty member at the University of Wisconsin—Madison and a Robert Wood Johnson Scholar in Health Policy Research at Harvard University. He has published numerous research articles in journals, including the *American Sociological Review*, *American Journal of Sociology*, and *Public Opinion Quarterly*. He has also taught categorical data analysis at the ICPSR Summer Program in Quantitative Methods and at El Colegio de México.

Telephone: 979-696-4600
800-782-8272
800-STATAPC
Fax: 979-696-4601

service@stata-press.com
stata-press.com

