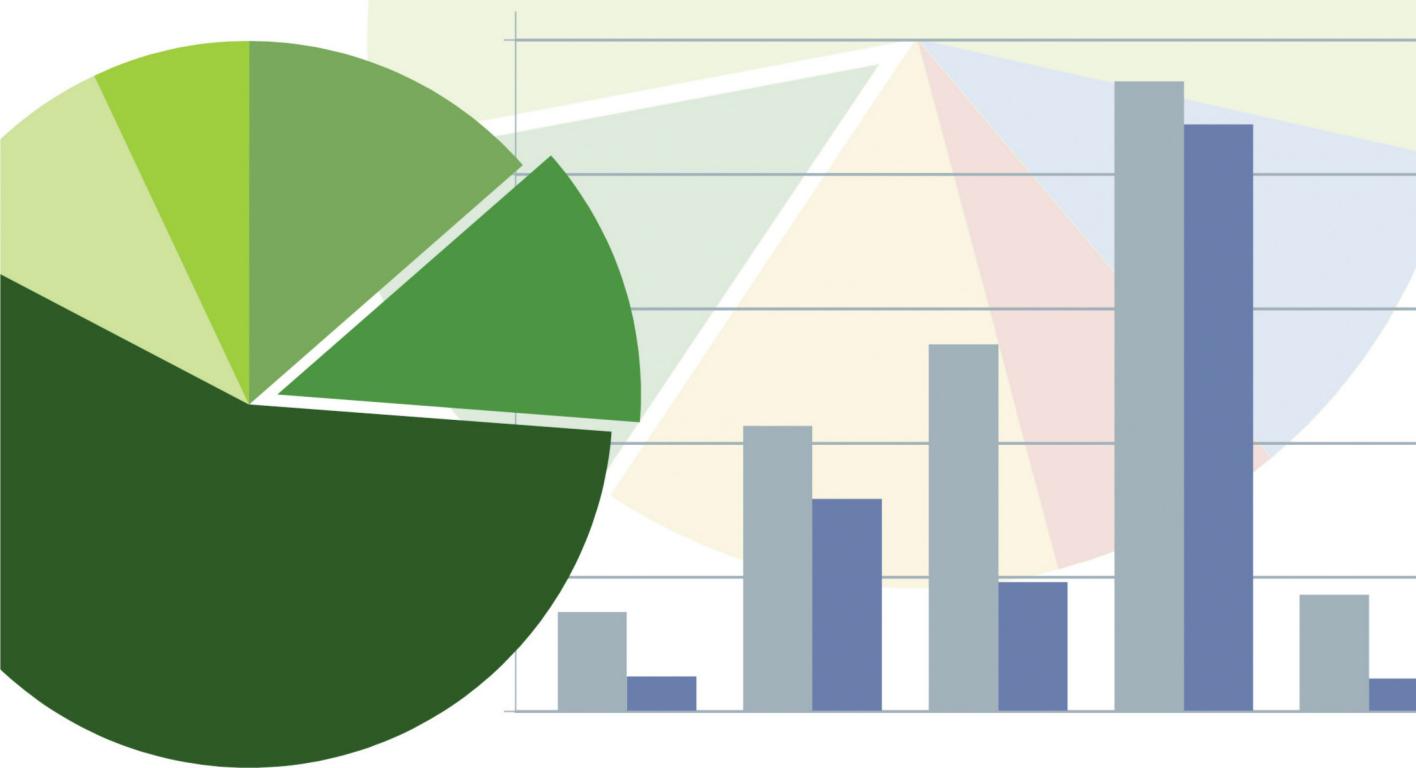


VERSION

12

Statistics with **STATA:**



Lawrence C. Hamilton

Statistics with STATA

Updated for Version 12

This is an electronic version of the print textbook. Due to electronic rights restrictions, some third party content may be suppressed. Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. The publisher reserves the right to remove content from this title at any time if subsequent rights restrictions require it. For valuable information on pricing, previous editions, changes to current editions, and alternate formats, please visit www.cengage.com/highered to search by ISBN#, author, title, or keyword for materials in your areas of interest.

Statistics

with

STATA

Updated for Version 12

Lawrence C. Hamilton
University of New Hampshire



Australia • Brazil • Japan • Korea • Mexico • Singapore • Spain • United Kingdom • United States

Statistics with STATA: Updated for Version 12

Eighth Edition

Lawrence C. Hamilton

Publisher/Executive Editor: Richard Stratton

Senior Sponsoring Editor: Molly Taylor

Assistant Editor: Shaylin Walsh Hogan

Editorial Assistant/Associate: Alexander Gontar

Media Editor: Andrew Coppola

Marketing Assistant: Lauren Beck

Marketing Communications Manager:

Jason LaChappelle

©2013, 2009, 2006 Brooks/Cole, Cengage Learning

ALL RIGHTS RESERVED. No part of this work covered by the copyright herein may be reproduced, transmitted, stored, or used in any form or by any means graphic, electronic, or mechanical, including but not limited to photocopying, recording, scanning, digitizing, taping, Web distribution, information networks, or information storage and retrieval systems, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the publisher.

For product information and technology assistance, contact us at
Cengage Learning Customer & Sales Support, 1-800-354-9706

For permission to use material from this text or product,
submit all requests online at www.cengage.com/permissions.

Further permissions questions can be emailed to
permissionrequest@cengage.com.

Library of Congress Control Number: 2012945319

ISBN-13: 978-0-8400-6463-9

ISBN-10: 0-8400-6463-2

Brooks/Cole

20 Channel Center Street

Boston, MA 02210

USA

Cengage Learning is a leading provider of customized learning solutions with office locations around the globe, including Singapore, the United Kingdom, Australia, Mexico, Brazil and Japan. Locate your local office at

international.cengage.com/region

Cengage Learning products are represented in Canada by Nelson Education, Ltd.

For your course and learning solutions, visit www.cengage.com.

Purchase any of our products at your local college store or at our preferred online store www.cengagebrain.com.

Instructors: Please visit login.cengage.com and log in to access instructor-specific resources.

Printed in the United States of America
1 2 3 4 5 6 7 16 15 14 13 12

Contents

Preface	ix
Notes on the Eighth Edition	x
Acknowledgments	xi
1 Stata and Stata Resources	1
A Typographical Note	1
An Example Stata Session	2
Stata's Documentation and Help Files	7
Searching for Information	8
StataCorp	9
The <i>Stata Journal</i>	10
Books Using Stata	11
2 Data Management	13
Example Commands	14
Creating a New Dataset by Typing in Data	16
Creating a New Dataset by Copy and Paste	21
Specifying Subsets of the Data: in and if Qualifiers	23
Generating and Replacing Variables	26
Missing Value Codes	29
Using Functions	32
Converting Between Numeric and String Formats	36
Creating New Categorical and Ordinal Variables	39
Using Explicit Subscripts with Variables	41
Importing Data from Other Programs	42
Combining Two or More Stata Files	46
Collapsing Data	49
Reshaping Data	52
Using Weights	55
Creating Random Data and Random Samples	57
Writing Programs for Data Management	61
3 Graphs	65
Example Commands	65
Histograms	68
Box Plots	71
Scatterplots and Overlays	74
Line Plots and Connected-Line Plots	80
Other Twoway Plot Types	85

Bar Charts and Pie Charts	87
Symmetry and Quantile Plots	90
Adding Text to Graphs	93
Graphing with Do-Files	95
Retrieving and Combining Graphs	96
Graph Editor	97
Creative Graphing	100
4 Survey Data	107
Example Commands	108
Declare Survey Data	108
Design Weights	110
Poststratification Weights	113
Survey-Weighted Tables and Graphs	115
Bar Charts for Multiple Comparisons	119
5 Summary Statistics and Tables	123
Example Commands	123
Summary Statistics for Measurement Variables	125
Exploratory Data Analysis	127
Normality Tests and Transformations	129
Frequency Tables and Two-Way Cross-Tabulations	133
Multiple Tables and Multi-Way Cross-Tabulations	136
Tables of Means, Medians and Other Summary Statistics	139
Using Frequency Weights	140
6 ANOVA and Other Comparison Methods	143
Example Commands	144
One-Sample Tests	145
Two-Sample Tests	148
One-Way Analysis of Variance (ANOVA)	151
Two- and N -Way Analysis of Variance	154
Factor Variables and Analysis of Covariance (ANCOVA)	155
Predicted Values and Error-Bar Charts	158
7 Linear Regression Analysis	163
Example Commands	163
Simple Regression	167
Correlation	170
Multiple Regression	174
Hypothesis Tests	179
Dummy Variables	181
Interaction Effects	185
Robust Estimates of Variance	190
Predicted Values and Residuals	192
Other Case Statistics	197
Diagnosing Multicollinearity and Heteroskedasticity	202

Confidence Bands in Simple Regression	205
Diagnostic Graphs	209
8 Advanced Regression Methods	215
Example Commands	215
Lowess Smoothing	217
Robust Regression	221
Further rreg and qreg Applications	227
Nonlinear Regression — 1	230
Nonlinear Regression — 2	233
Box–Cox Regression	238
Multiple Imputation of Missing Values	241
Structural Equation Modeling	244
9 Logistic Regression	251
Example Commands	252
Space Shuttle Data	254
Using Logistic Regression	258
Marginal or Conditional Effects Plots	262
Diagnostic Statistics and Plots	264
Logistic Regression with Ordered-Category y	268
Multinomial Logistic Regression	270
Multiple Imputation of Missing Values — Logit Regression Example	278
10 Survival and Event-Count Models	283
Example Commands	284
Survival-Time Data	286
Count-Time Data	288
Kaplan–Meier Survivor Functions	290
Cox Proportional Hazard Models	293
Exponential and Weibull Regression	299
Poisson Regression	303
Generalized Linear Models	307
11 Principal Component, Factor and Cluster Analysis	313
Example Commands	314
Principal Component Analysis and Principal Component Factoring	315
Rotation	318
Factor Scores	321
Principal Factoring	323
Maximum-Likelihood Factoring	325
Cluster Analysis — 1	327
Cluster Analysis — 2	331
Using Factor Scores in Regression	336
Measurement and Structural Equation Models	344

12 Time Series Analysis	351
Example Commands	351
Smoothing	353
Further Time Plot Examples	359
Recent Climate Change	363
Lags, Lead and Differences	366
Correlograms	371
ARIMA Models	374
ARMAX Models	382
13 Multilevel and Mixed-Effects Modeling	387
Example Commands	388
Regression with Random Intercepts	390
Random Intercepts and Slopes	395
Multiple Random Slopes	400
Nested Levels	404
Repeated Measurements	406
Cross-Sectional Time Series	410
Mixed-Effects Logit Regression	415
14 Introduction to Programming	423
Basic Concepts and Tools	423
Do-files	423
Ado-files	424
Programs	425
Local macros	426
Global macros	427
Scalars	427
Version	427
Comments	428
Looping	429
If ... else	430
Arguments	431
Syntax	432
Example Program: multicat (Plot Many Categorical Variables)	434
Using multicat	437
Help File	441
Monte Carlo Simulation	445
Matrix Programming with Mata	452
Dataset Sources	457
References	461
Index	469

Preface

Statistics with Stata is intended for students and practicing researchers, to bridge the gap between statistical textbooks and Stata's own documentation. In this intermediate role, it does not provide the detailed expositions of a proper textbook, nor does it come close to describing all of Stata's features. Instead, it demonstrates how to use Stata to accomplish a wide variety of statistical tasks. Chapter topics follow conceptual themes rather than focusing on particular Stata commands, which gives *Statistics with Stata* a different structure from the Stata reference manuals. The chapter on Data Management, for example, covers many different procedures for creating, importing, combining or restructuring data files. Chapters on Graphs, Summary Statistics and Tables, and on ANOVA and Other Comparison Methods have similarly broad themes that encompass a number of separate techniques. A new chapter that introduces Survey Data, placed early in the book, provides background for more technical survey examples presented where appropriate in later chapters.

The general topics of the first seven chapters (through Linear Regression Analysis) roughly parallel an undergraduate or first graduate-level course in applied statistics, but with additional depth to cover practical issues often encountered by analysts — how to import datasets, draw publication-quality graphics, work with survey weights, or do trouble-shooting in regression, for instance. In Chapter 8 (Advanced Regression Methods) and beyond, we move into the territory of advanced courses or original research. Here, readers can find basic information and illustrations of lowess, robust, quantile, nonlinear, logit, ordered logit, multinomial logit or Poisson regression; apply new methods for structural equation modeling or multiple imputation of missing values; fit survival-time and event-count models; construct and use composite variables from factor analysis or principal components; divide observations into empirical types or clusters; analyze simple or multiple time series; and fit multilevel or mixed-effects models. Stata has worked hard in recent years to advance its state-of-the-art standing, and this effort is particularly apparent in the wide range of statistical modeling commands it now offers.

The book concludes with a look at programming in Stata. Many readers will find that Stata does everything they need already, so they have no reason to write original programs. For an active minority, however, programmability is one of Stata's principal attractions, and it underlies Stata's currency and rapid advancement. Chapter 14 opens the door for new users to explore Stata programming, whether for specialized data management tasks, to establish a new statistical capability, for Monte Carlo experiments or for teaching.

Generally similar versions ("flavors") of Stata run on Windows, Mac and Unix computers. Across all platforms, Stata uses the same commands and produces the same output. Datasets, graphs and programs created on one platform can be used by Stata running on any other platform. The flavors differ in some details of screen appearance, menus and file handling, where Stata follows the conventions native to each platform — such as \directory\filename file specifications under Windows, in contrast with the /directory/filename specifications under

Unix. Rather than display all three, I employ Windows conventions, but users with other systems should find that only minor translations are needed.

Notes on the Eighth Edition

I began using Stata in 1985, the first year of its release. Initially, Stata ran only on MS-DOS personal computers, but its desktop orientation made it distinctly more modern than its main competitors — most of which had originated before the desktop revolution, in the 80-column punched-card Fortran environment of mainframes. Unlike mainframe statistical packages that believed each user was a stack of cards, Stata viewed the user as a conversation. Its interactive nature and integration of statistical procedures with data management and graphics supported the natural flow of analytical thought in ways that other programs did not. **graph** and **predict** soon became favorite commands. I was impressed enough by how it all fit together to start writing the first external Stata book, *Statistics with Stata*, published in 1989 for Stata version 2. Stata's 20th anniversary in 2005 was marked by a special issue of the *Stata Journal*, filled with historical articles, interviews and by invitation a brief history of *Statistics with Stata*.

A great deal about Stata has changed since this book's first edition, in which I observed that "Stata is not a do-everything program The things it does, however, it does very well." The expansion of Stata's capabilities has been striking. This is very noticeable in the proliferation, and later in the steady rationalization, of model fitting procedures. William Gould's architecture for Stata, with its programming tools and unified syntax, has aged well and smoothly incorporated new statistical methods as these were developed. The broad range of graphs in Chapter 3, the formidable list of modeling commands that begins Chapter 8, or the new time series, survey, multiple-imputation or mixed-modeling capabilities discussed in later chapters illustrate some of the ways that Stata became richer over the years. Suites of new techniques such as those for panel (**xt**), survey (**svy**), time series (**ts**), survival time (**st**) or multiple imputation (**mi**) data open worlds of possibility, as do programmable commands for generalized linear modeling (**glm**), or general procedures for maximum-likelihood estimation. Other major extensions include the development of a matrix programming capability, the wealth of new data-management features, and new multipurpose analytical tools such as marginal plots or structural equation modeling. Data management, with good reason, has been promoted from an incidental topic in the first *Statistics with Stata* to the longest chapter in this eighth edition.

Stata's extensive menu and dialog-box system provides point-and-click alternatives to most typed commands. Series of menu and dialog selections are easier to learn through exploration than through reading, however, so *Statistics with Stata* provides only general suggestions about menus at the beginning of each chapter. For the most part, I employ commands to show what Stata can do; those commands' menu counterparts should be easy to discover. Conversely, if you start out working mainly through menus, Stata provides informal training by showing each corresponding command in the Results window. The menu/dialog system works by translating clicks into Stata commands, which it then feeds to Stata for execution.

Analytical graphics are a great strength of Stata, as displayed throughout every chapter. Many of my examples are not bare-bones images meant to demonstrate one particular technique, but incorporate some enhancements toward publication or presentation quality. Readers might browse the figures for ideas about graphical possibilities, beyond what appears in Stata manuals.

Statistics with Stata version 12 differs substantially from the book's version 10 predecessor. Chapters have been reorganized, including a new introductory Survey Data chapter that comes early in the book. Regression topics from four chapters of the version 10 book have been integrated and organized more logically into two longer chapters here, on Linear Regression and Advanced Regression Methods. The Advanced Regression chapter contains new sections on multiple imputation of missing values and on structural equation modeling (SEM). The Principal Component, Factor and Cluster Analysis chapter includes two new sections as well, showing the use of factor scores in regression, and the use of measurement models in SEM. A new section in the Multilevel and Mixed-Effects Modeling chapter presents a repeated-measures experiment. The final chapter on programming has been streamlined and centered around a main example (draw multiple survey graphs) that could prove useful to some readers.

One goal for this version 12 revision was to upgrade many of the examples, some of which dealt with my research from the 1990s but had outlived their charm. The *Challenger* space shuttle analysis, featured on the original 1989 edition cover, still works well to present basic ideas at the start of the Logistic Regression chapter. That chapter now ends, however, with a weighted multinomial logit analysis of responses to a 2011 survey asking what people know and believe about climate change. The climate survey is one of three new 2010 or 2011 survey datasets that provide key examples across several chapters. One such chapter (Principal Component and Factor Analysis) begins with a simple planetary dataset, but ends with new sections on combining factor analysis with regression, or the analogous measurement and structural equation models, using a 2011 coastal-environment survey. Other running examples involve time series of physical climate indicators. One unique dataset on 42 Arctic Alaska villages, drawn from a 2011 paper, illustrates how mixed-effects modeling can integrate natural with social science data. The ARMAX models wrapping up the Time Series chapter are inspired by an influential 2011 paper that investigated the “real signal” of global warming. Where possible, I aim for examples that pose research questions of general interest, rather than just supplying numbers to illustrate a technique. Many example datasets include other variables beyond those discussed in the text, inviting readers to do further analysis on their own.

As noted in Chapter 1, Stata's help and search features have advanced to keep pace with the program. Behind the interactive documentation available through help files stand Stata's website, Internet and documentation search capabilities, user-community listserv, NetCourses, the *Stata Journal*, and over 9,000 pages of documentation. *Statistics with Stata* provides an accessible gateway to Stata; these other resources will help you go further.

Acknowledgments

Stata's architect, William Gould, deserves credit for originating the elegant program that *Statistics with Stata* describes. Many others at StataCorp contributed their insights and advice over the years. For this eighth edition I am particularly grateful to Pat Branton, who organized the reviews, and Kristin MacDonald who read most of the chapters. James Hamilton gave key advice about time series for Chapters 12 and 13. Leslie Hamilton read and helped to edit many parts of the final manuscript.

The book is built around data. A new section in this edition provides notes on dataset sources, including Internet links if these exist, or citations to published articles. Many examples come from public sources that are products of other researchers' hard work. I also drew on my own research, particularly some recent surveys, and studies that integrate natural with social-science data. All of the colleagues who worked on these projects with me deserve a share of the credit, including Mil Duncan and Tom Safford (CERA rural surveys); Richard Lammers, Dan White and Greta Myerchin (Alaska communities); David Moore and Cameron Wake (climate surveys); Barry Keim and Cliff Brown (skiing and climate studies); and Rasmus Ole Rasmussen and Per Lyster Pedersen (Greenland demographics). Others who generously shared their original data include Dave Hamilton, Dave Meeker, Steve Selvin, Andrew Smith and Sally Ward.

Dedication

To Leslie, Sarah and Dave.

Stata and Stata Resources

Stata is a full-featured statistical program for Windows, Mac and Unix computers. It combines ease of use with speed, a library of pre-programmed analytical and data-management capabilities, and programmability that allows users to invent and add further capabilities as needed. Most operations can be accomplished either via the pull-down menu system, or more directly via typed commands. Menus help newcomers to learn Stata, and help anyone to apply an unfamiliar procedure. The consistent, intuitive syntax of Stata commands frees experienced users to work more efficiently, and also makes it straightforward to develop programs for complex or repetitious tasks. Menu and command instructions can be mixed as needed during a Stata session. Extensive help, search and link features make it easy to look up command syntax and other information instantly, on the fly. This book is written to complement those features.

After introductory information, we will begin with an example Stata session to give you a sense of the flow of data analysis, and how analytical results might be used. Later chapters explain in more detail. Even without explanations, however, you can see how straightforward the commands are — **use *filename*** to retrieve dataset *filename*, **summarize** when you want summary statistics, **correlate** to get a correlation matrix, and so forth. Alternatively, the same results can be obtained by making choices from the Data or Statistics menus.

Stata users have available a variety of resources to help them learn about Stata and solve problems at any level of difficulty. These resources come not just from StataCorp, but also from an active community of users. Sections of this chapter introduce some key resources — Stata's online help and printed documentation; where to write or e-mail for technical help; Stata's website (www.stata.com), which provides many services including updates and answers to frequently asked questions; the Statalist Internet list; and the refereed *Stata Journal*.

A Typographical Note

This book employs several typographical conventions as a visual cue to how words are used:

- Commands typed by the user appear in **bold**. When the whole command line is given, it starts with a period, as seen in a Stata Results window or log (output) file:

```
. correlate extent area volume temp
```

2 Statistics with Stata

- *Variable* or *file* names within these commands appear in italics to emphasize the fact that they are arbitrary and not a fixed part of the command.
- Names of *variables* or *files* also appear in italics within the main text to distinguish them from ordinary words.
- Items from Stata's menus are shown in the Arial font, with successive options separated by “>”. For example, we can open an existing dataset by selecting File > Open , and then finding and clicking on the name of the particular dataset. Some common menu actions can be accomplished either with text choices from Stata's top menu bar,

File Edit Data Graphics Statistics User Window Help

or with the row of icons below these. For example, selecting File > Open is equivalent to clicking the leftmost icon, a tiny picture of an opening file folder  One could also accomplish the same thing by typing a direct command of the form

. use *filename*

Thus, we show the calculation of summary statistics for a variable named *extent* as follows:

. summarize extent

Variable	Obs	Mean	Std. Dev.	Min	Max
extent	33	6.51697	.9691796	4.3	7.88

These typographic conventions exist only in this book, and not within the Stata program itself. Stata can display a variety of onscreen fonts, but it does not use italics in commands. Once Stata log files have been imported into a word processor, or a results table has been copied and pasted, you might want to format them in a Courier font, 10 point or smaller, so that columns will line up correctly.

In its commands and variable names, Stata is case sensitive. Thus, **summarize** is a command, but Summarize and SUMMARIZE are not. *Extent* and *extent* would be two different variables.

An Example Stata Session

As a preview showing Stata at work, this section retrieves and analyzes a previously-created dataset named *Arctic9.dta*. This small time series covers satellite-era (1979 to 2011) observations of ice on the Arctic Ocean in September, at the lowest point of its annual cycle. The data come from three different sources (see the appendix on Data Sources). One variable, *extent*, is a satellite-based measure of the Northern Hemisphere sea area with at least 15% ice concentration each September. *Area* numbers are somewhat less than *extent*, representing the area of sea ice itself. Another variable, *tempN*, describes mean annual surface air temperature above 64°N latitude. Temperatures are expressed as anomalies, which are deviations from the 1951–1980 average, in degrees Celsius. We have 33 observations (years) and 8 variables.

If we might eventually want a record of our session, the best way to prepare for this is by opening a log file at the start. Log files contain commands and results tables, but not graphs. To begin a log file, choose **File > Log > Begin ...** from the top menu bar, and specify a name and folder for the resulting log file. Alternatively, a log file could be started by choosing **File > Log > Begin** from the top menu bar, or by typing a direct command such as

```
. log using monday1
```

Multiple ways of doing such things are common in Stata. Each way has its own advantages, and each suits different situations or user tastes.

Log files can be created either in a special Stata format (.smcl), or in ordinary text or ASCII format (.log). A .smcl (Stata markup and control language) file will be nicely formatted for viewing or printing within Stata. It could also contain hyperlinks that help to understand commands or error messages. .log (text) files lack such formatting, but are simpler to use if you plan later to insert or edit the output in a word processor. After selecting which type of log file you want, click **Save**. For this session, we will create a .smcl log file named *monday1.smcl*.

An existing Stata-format dataset named *Arctic9.dta* will be analyzed here. To open or retrieve this dataset, we again have several options:

select **File > Open > Arctic9.dta** using the top menu bar;

click on  > *Arctic9.dta*; or

type the command **use Arctic9**.

Under its default Windows configuration, Stata looks for data files in the user's Documents directory. If the file we want is in a different folder, we could specify its location in the **use** command,

```
. use C:\books\sws_12\data\Arctic9
```

or change the session's default folder by issuing a **cd** (change directory) command,

```
. cd C:\books\sws_12\data\  
. use Arctic9
```

or select **File > Change Working Directory ...** from the menus. Often, the simplest way to retrieve a file will be to choose **File > Open** and browse through folders in the usual way.

To see a brief description of the dataset now in memory, type

```
. describe
```

Contains data from C:\data\Arctic9.dta				Arctic September mean sea ice
obs:	33 <th data-cs="3" data-kind="parent">1979-2011 17 Apr 2012 09:21</th> <th data-kind="ghost"></th> <th data-kind="ghost"></th>	1979-2011 17 Apr 2012 09:21		
vars:	8			
size:	891			
variable	storage type	display format	value label	variable label
year	int	%ty		Year
month	byte	%8.0g		Month
extent	float	%9.0g		Sea ice extent, million km ²
area	float	%9.0g		Sea ice area, million km ²
volume	float	%8.0g		Sea ice volume, 1000 km ³
volumehi	float	%9.0g		Volume + 1.35 (uncertainty)
volumelo	float	%9.0g		Volume - 1.35 (uncertainty)
tempN	float	%9.0g		Annual air temp anomaly 64N-90N C

Sorted by: year

Many Stata commands can be abbreviated to their first few letters. For example, we could shorten **describe** to just the letter **d**. Using menus, the same table could be obtained by choosing

Data > Describe data > Describe data in memory > (OK).

This dataset has only 33 observations and 8 variables, so we could list all its contents by typing the command **list** (or the letter **l**; or Data > Describe data > List data > (OK)). To save space here we list only the first 10 years, typing **list in 1/10**:

```
. list in 1/10
```

	year	month	extent	area	volume	volumehi	volumelo	tempN
1.	1979	9	7.2	5.72	16.9095	18.2595	15.5595	-.57
2.	1980	9	7.85	6.02	16.3194	17.66937	14.96937	.33
3.	1981	9	7.25	5.57	12.8131	14.16307	11.46307	1.21
4.	1982	9	7.45	5.57	13.5099	14.85987	12.15987	-.34
5.	1983	9	7.52	5.83	15.2013	16.5513	13.8513	.27
6.	1984	9	7.17	5.24	14.6336	15.98357	13.28357	.31
7.	1985	9	6.93	5.36	14.5836	15.93363	13.23363	.3
8.	1986	9	7.54	5.85	16.0803	17.43027	14.73027	-.05
9.	1987	9	7.48	5.91	15.3609	16.7109	14.0109	-.25
10.	1988	9	7.49	5.62	14.988	16.338	13.638	.87

Analysis could begin with a table of means, standard deviations, minimum values, and maximum values. Type **summarize** or **su**; or select from the drop-down menus, Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Summary statistics > (OK)

```
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
year	33	1995	9.66954	1979	2011
month	33	9	0	9	9
extent	33	6.51697	.9691796	4.3	7.88
area	33	4.850303	.8468452	3.09	6.02
volume	33	12.04664	3.346079	4.210367	16.9095
volumehi	33	13.39664	3.346079	5.560367	18.2595
volumelo	33	10.69664	3.346079	2.860367	15.5595
tempN	33	.790303	.7157928	-.57	2.22

To print results from the session so far, click on the Results window and then , or from the menus choose File > Print > Results .

To copy a table, commands, or other information from the Results window into a word processor, drag the mouse to select the results you want, right-click the mouse, and then choose Copy Text from the mouse's menu. Switch to your word processor and, at the desired insertion point either right-click and Paste or click the word processor's paste icon. A final step in most cases will be to change the pasted text to a fixed-width font such as Courier.

Arctic sea ice extent, area and volume should be related to annual air temperature, not only because warmer air contributes to ice melting but also because surface air temperatures over ice-free seas will be warmer than temperatures over ice. We can see the correlations among variables by typing **correlate** followed by a list of variables.

```
. correlate extent area volume tempN
(obs=33)
```

	extent	area	volume	tempN
extent	1.0000			
area	0.9826	1.0000		
volume	0.9308	0.9450	1.0000	
tempN	-0.8045	-0.8180	-0.8651	1.0000

September sea ice *extent*, *area* and *volume* all have strong positive correlations, as one might expect. Their correlation with annual air temperature is negative: the warmer the air, the less ice (or vice versa). The same correlation matrix could be obtained through menus:

Statistics > Summaries, tables, and tests > Summary and descriptive statistics > Correlation and covariance

Then choose the variables to be correlated. Although menu choices often are straightforward to use, you can see that they are more complicated to describe than the simple text commands. From this point on, we will focus primarily on the commands, mentioning menu alternatives only occasionally. Fully exploring the menus, and working out how to use them to accomplish the same tasks, will be left to the reader. For similar reasons, the Stata reference manuals likewise take a command-based approach.

So ice extent, area, volume and temperature all are related. How have they changed over time? Figure 1.1 plots *extent* against *year*, produced by the **graph twoway connect** command. The first-named variable in this command, *extent*, defines the vertical or *y* axis; the last-named

variable, *year*, defines the horizontal or *x* axis. We see an uneven but steepening downward pattern, as September sea ice extent declined by more than a third over this period.

```
. graph twoway connect extent year
```

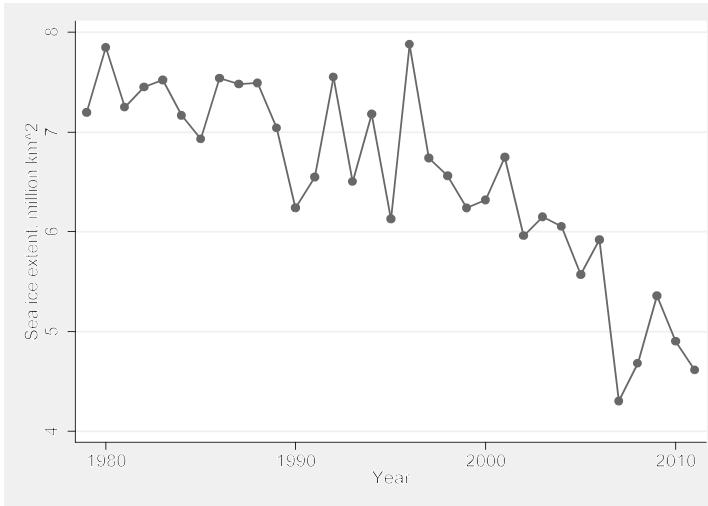


Figure 1.1

To print this graph, go to the Graph window and click its print icon or File > Print. To copy the graph directly into a word processor or other document, right-click on the graph, and select Copy Graph. Switch to your word processor, go to the desired insertion point, and issue an appropriate paste command such as Edit > Paste, Edit > Paste Special (Metafile), or click a paste icon (different word processors will handle this differently).

To save the graph for future use, either right-click and Save Graph, click in the Graph window, or select File > Save As from the Graph window's top menu bar. The Save as type submenu offers several different file formats. On a Windows system, the choices include

- Stata graph (*.gph) (A “live” graph, containing enough information for Stata to edit)
- As-is graph (*.gph) (A more compact Stata graph format)
- Windows Metafile (*.wmf)
- Enhanced Metafile (*.emf)
- Portable Network Graphics (*.png)
- TIFF (*.tif)
- PostScript (*.ps)
- Encapsulated PostScript with or without TIFF preview (*.eps)
- Portable Document File (*.pdf)

Other platforms such as Mac or Linux offer different choices for graph file formats. Regardless of which format we want, it often is worthwhile to save one copy of our graph in live .gph format. Such live .gph-format graphs can later be retrieved, combined, recolored or reformatted

using the **graph use** or **graph combine** commands, or edited using the Graph Editor (Chapter 3).

Through all of the preceding analyses, the log file *monday1.smcl* has been storing our results. An easy way to review this file to see what we have done is to open the file in its own Viewer window by selecting

File > Log > View > OK

We could print this log file by clicking the  icon on the top bar of the log file's Viewer window. Log files close automatically at the end of a Stata session, or earlier if instructed by  > Close log file, typing the command **log close**, or by choosing

File > Log > Close

Once closed, the file *monday1.smcl* could be opened to view again through File > Log > View or  during a subsequent Stata session. To create an output file that can be opened easily by your word processor, either translate the log file from .smcl (a Stata format) to .log (standard ASCII text format) by typing

. translate monday1.smcl monday1.log

or start out by creating the file in .log instead of .smcl format. You can also start and stop a log file temporarily, any number of times:

File > Log > Suspend

File > Log > Resume

The log icon  on Stata's main icon menu bar can also perform all these tasks.

Stata's Documentation and Help Files

The complete Stata 12 Documentation Set includes 19 volumes: a slim *Getting Started* manual (for example, *Getting Started with Stata for Windows*), the more extensive *User's Guide*, the encyclopedic four-volume *Base Reference Manual*, and separate reference manuals on data management, graphics, longitudinal and panel data, matrix programming (Mata), multiple imputation, multivariate statistics, programming, structural equation modeling, survey data, survival analysis and epidemiological tables, and time series analysis. *Getting Started* helps you do just that, with the basics of installation, window management, data entry, printing, and so on. The *User's Guide* contains an extended discussion of general topics, including resources and troubleshooting. Of particular note for new users is the *User's Guide* section on “Commands everyone should know.” The *Base Reference Manual* lists all Stata commands alphabetically. Entries for each command include the full command syntax, descriptions of all available options, examples, technical notes regarding formulas and rationale, and references for further reading. Data management, graphics, panel data etc. are covered in the general references, but

these complicated topics get more detailed treatment and examples in their own specialized manuals. A *Quick Reference and Index* volume rounds out the whole collection. Although the physical manuals fill a bookshelf, complete PDFs can be accessed within Stata at any time through Help > PDF Documentation, or through links if you type **help** followed by a specific command name.

When we are in the midst of a Stata session, it is easy to ask for onscreen help, which in turn can connect with the manuals. Selecting Help from the top menu bar invokes a drop-down menu of further choices, including specific commands, what's new, online updates, the *Stata Journal* and user-written programs, or connections to Stata's website (www.stata.com). Choosing Search allows keyword searching of Stata's documentation, of Net resources, or both. Alternatively, choosing Contents (or typing **help**) allows us to look up how to do things by category. The **help** command is particularly useful when used with a command name. Typing **help correlate**, for example, causes a description of that command to appear in a Viewer window. Like the reference manuals, this onscreen help provides command syntax diagrams and complete lists of options. It also includes some examples, although often less detailed and without the technical discussions found in the manuals. The onscreen help has several advantages over the manuals, however. The Viewer allows searching for keywords in the documentation or on Stata's website. Hypertext links take you directly to related entries. Onscreen help can also include material about recent updates, or the unofficial Stata programs that you have downloaded from Stata's website or from other users.

Searching for Information

Selecting Help > Search > Search documentation and FAQs provides a direct way to search for information in Stata's documentation or in the website's FAQs (frequently asked questions) and other pages. Alternatively, we can search net resources including the *Stata Journal*. Search results in the Viewer window contain clickable hyperlinks leading to further information or original citations.

The **search** command can do similar things. One specialized use for a quick **search** command is to provide more information on those occasions when our command does not succeed as planned, but instead results in one of Stata's cryptic numerical error messages. For example, **table** is a Stata command, but it requires information about what exactly we want in our table. If we mistakenly type **table** by itself, Stata responds with the error message and cryptic "return code" r(100):

```
. table
varlist required
r(100);
```

Clicking on the return code r(100) in this error message brings up a more informative note. We could also find this note by typing **search rc 100**. Type **help search** for more about this command.

StataCorp

The mailing or physical address is

StataCorp
4905 Lakeway Drive
College Station, TX 77845 USA

Telephone access includes an easy-to-remember 800 number.

telephone:	1-800-782-8272	(or 1-800-STATAPC) U.S.
	1-800-248-8272	Canada
	1-979-696-4600	other International
fax:	1-979-696-4601	

For orders, licensing, and upgrade information, you can contact StataCorp by e-mail at

service@stata.com

or visit their website at

<http://www.stata.com>

Stata Press also has its own website, containing information about Stata publications including the datasets used for examples.

<http://www.stata-press.com>

The refereed *Stata Journal* has become an important resource as well.

<http://www.stata-journal.com>

Stata's main website, www.stata.com, provides extensive user resources, starting with pages describing Stata products in detail, how to order Stata, and many kinds of user support such as:

FAQs — Frequently asked questions and their answers. If you are puzzled by something and can't find the answer in the manuals, check here next — it might be a FAQ. Example questions range from basic questions such as "How can I convert other packages' files to Stata format data files?" to more technical queries like "How do I impose the restriction that rho is zero using the heckman command with full ml?"

Updates — Online updates within major versions are free to registered Stata users. These provide a fast, simple way to obtain the latest enhancements, bug fixes, etc. for your current version. Instead of going to the website you can ask within Stata whether updates exist for your version, and initiate the update process by typing the command

. update query

Technical support — Technical support can be obtained by sending e-mail messages to

tech-support@stata.com

Responses tend to be prompt and helpful. Before writing for technical help, though, you should check whether your question is a FAQ.

Training — Enroll in web-based NetCourses on selected topics such as Introduction to Stata, Introduction to Stata Programming, or Advanced Stata Programming.

Stata News — The *Stata News* contains information about software features, current NetCourses, recent issues of the *Stata Journal*, and other topics.

Publications — Links to information about the *Stata Journal*, documentation and manuals, a bookstore selling books about Stata and other up-to-date statistical references, and Stata's author support program for people writing new books about Stata. The following sections have more to say about the *Stata Journal* and Stata books.

Stata's website hosts The Stata Blog,
<http://blog.stata.com/>

Users of social media might also find it entertaining and informative to follow Stata on Twitter (www.twitter.com) or like Stata on Facebook (www.facebook.com).

The Stata Journal

From 1991 through 2001, a bimonthly publication called the *Stata Technical Bulletin* (*STB*) served as a means of distributing new commands and Stata updates, both user-written and official. Accumulated *STB* articles were published in book form each year as *Stata Technical Bulletin Reprints*, which can be ordered directly from StataCorp. With the growth of the Internet, instant communication among users became possible. Program files could easily be downloaded from distant sources. A bimonthly printed journal and disk no longer provided the best avenues either for communicating among users, or for distributing updates and user-written programs. To adapt to a changing world, the *STB* had to evolve into something new.

The *Stata Journal* was launched to meet this challenge and the needs of Stata's broadening user base. Like the old *STB*, the *Stata Journal* contains articles describing new commands by users along with unofficial commands written by StataCorp employees. New commands are not its primary focus, however. The *Stata Journal* also contains refereed expository articles about statistics, book reviews, tips on using Stata, and a number of interesting columns, including Speaking Stata by Nicholas J. Cox, on effective use of the Stata programming language. The *Stata Journal* is intended for novice as well as experienced Stata users. For example, here are the contents from the June 2012 issue.

Articles and columns

- “A robust instrumental-variables estimator,” R. Desbordes, V. Verardi
- “What hypotheses do ‘nonparametric’ two-group tests actually test?” R.M. Conroy
- “From resultsets to resultstable in Stata,” R.B. Newson
- “Menu-driven X-12-ARIMA seasonal adjustment in Stata,” Q. Wang, N. Wu
- “Faster estimation of a discrete-time proportional hazards model with gamma frailty,” M.G. Farnworth
- “Threshold regression for time-to-event analysis: The stthreg package,” T. Xiao, G.A. Whitmore, X. He, M.-L.T. Lee

“Fitting nonparametric mixed logit models via expectation-maximization algorithm,” D. Pacifico

“The S-estimator of multivariate location and scatter in Stata,” V. Verardi, A. McCathie

“Using the margins command to estimate and interpret adjusted predictions and marginal effects,” R. Williams

“Speaking Stata: Transforming the time axis,” N.J. Cox

Notes and Comments

“Stata tip 108: On adding and constraining,” M.L. Buis

“Stata tip 109: How to combine variables with missing values,” P.A. Lachenbruch

“Stata tip 110: How to get the optimal k-means cluster solution,” A. Makles

Software Updates

The *Stata Journal* is published quarterly. Subscriptions can be purchased by visiting www.stata-journal.com. The www.stata-journal.com archives list contents of back issues, which you can order individually; articles three years old or more can be downloaded for free. Of historical interest, a special issue on the occasion of Stata’s 20th anniversary (5(1), 2005) contains articles about the early development of Stata, and one about the first Stata book: “A short history of *Statistics with Stata*.”

Books Using Stata

In addition to Stata’s own reference manuals, a growing library of books describe Stata, or use Stata to illustrate analytical techniques. These books include general introductions; disciplinary applications such as social science, biostatistics, or econometrics; and focused texts concerning survey analysis, experimental data, categorical dependent variables, and other subjects.

The Bookstore pages on Stata’s website have up-to-date lists, with descriptions of content:

<http://www.stata.com/bookstore/>

This online bookstore provides a central place to learn about and order Stata-relevant books from many different publishers. Examples below illustrate the wide range of choices.

A Gentle Introduction to Stata, A.C. Acock

Using Stata for Principles of Econometrics, L.C. Adkins, R.C. Hill

An Introduction to Modern Econometrics Using Stata, C.F. Baum

Applied Microeconometrics Using Stata, A.C. Cameron, P.K. Trivedi

Event History Analysis with Stata, H-P. Blossfeld, K. Golsch, G. Rohwer

An Introduction to Survival Analysis Using Stata, M. Cleves, W. Gould, R. Gutierrez, Y.

Marchenko

Statistical Modeling for Biomedical Researchers, W.D. Dupont

Maximum Likelihood Estimation with Stata, W. Gould, J. Pitblado, B. Poi

Statistics with Stata, L.C. Hamilton

Generalized Linear Models and Extensions, J.W. Hardin, J.M. Hilbe

Negative Binomial Regression, J.M. Hilbe

A Short Introduction to Stata for Biostatistics, M. Hills, B.L. De Stavola

- Applied Survival Analysis: Regression Modeling of Time to Event Data*, D.W. Hosmer, S. Lemeshow, S. May
- Applied Econometrics for Health Economists*, A. Jones
- Applied Health Economics*, A. Jones, N. Rice, T.B. d'Uva, S. Balia
- An Introduction to Stata for Health Researchers*, S. Juul, M. Frydenberg
- Data Analysis Using Stata*, U. Kohler, F. Kreuter
- Sampling of Populations: Methods and Applications*, P.S. Levy, S. Lemeshow
- Workflow in Data Analysis Using Stata*, J.S. Long
- Regression Models for Categorical Dependent Variables Using Stata*, J.S. Long, J. Freese
- A Visual Guide to Stata Graphics*, M. Mitchell
- Data Management Using Stata: A Practical Handbook*, M. Mitchell
- Interpreting and Visualizing Regression Models Using Stata*, M. Mitchell
- Seventy-six Stata Tips*, H.J. Newton, N. J. Cox editors
- Analyzing Health Equity Using Household Survey Data*, O. O'Donnell and others
- A Stata Companion to Political Analysis*, P.H. Pollock III
- A Handbook of Statistical Analyses Using Stata*, S. Rabe-Hesketh, B. Everitt
- Multilevel and Longitudinal Modeling Using Stata*, S. Rabe-Hesketh, A. Skrondal
- Managing Your Patients? Data in the Neonatal and Pediatric ICU*, J. Schulman
- Epidemiology: Study Design and Data Analysis*, M. Woodward

Data Management

The first steps in data analysis involve organizing the raw data into a format usable by Stata. We can bring new data into Stata in several ways: type the data from the keyboard; import the data from another program such as Microsoft Excel; read a text or ASCII file containing the raw data; paste data from a spreadsheet into the Editor; or, using a third-party data transfer program, translate the dataset directly from a system file created by another spreadsheet, database or statistical program. Once Stata has the data in memory, we can save the data in Stata format for easy retrieval and updating in the future.

Data management encompasses the initial tasks of creating a dataset, editing to correct errors, identifying the missing values, and adding internal documentation such as variable and value labels. It also encompasses many other jobs required by ongoing projects, such as adding new observations or variables; reorganizing, simplifying or sampling from the data; separating, combining or collapsing datasets; converting variable types; and creating new variables through algebraic or logical expressions. When data-management tasks become repetitive or complex, Stata users can write their own programs to automate the work. Although Stata is best known for its analytical capabilities, it possesses a broad range of data-management features as well. This chapter introduces some of the basics.

The *User's Guide* provides an overview of the different methods for inputting data, followed by nine rules for determining which input method to use. Input, editing and many other operations discussed in this chapter can be accomplished through the Data menus. Data menu subheadings refer to the general category of task:

- Describe data
- Data Editor
- Create or change data
- Variables Manager
- Data utilities
- Sort
- Combine datasets
- Matrices, Mata language
- Matrices, ado language
- Other utilities

Example Commands

- **append using olddata**

Reads previously-saved dataset *olddata.dta* and adds all its observations to the data currently in memory. Subsequently typing **save newdata, replace** will save the combined dataset as *newdata.dta*.

- **browse**

Opens the spreadsheet-like Data Browser for viewing the data. The Browser looks similar to the Data Editor (see below), but it has no editing capability, so there is no risk of inadvertently changing your data. Alternatively, use the Data menu or click 

- **browse year month extent if year > 1999**

Opens the Data Browser showing only the variables *year*, *month* and *extent* for observations in which *year* is greater than 1999. This example illustrates the **if** qualifier, which can be used to focus the operation of many Stata commands.

- **compress**

Automatically converts all variables to their most efficient storage types to conserve memory and disk space. Subsequently typing the command **save filename, replace** will make these changes permanent.

- **drawnorm z1 z2 z3, n(5000)**

Creates an artificial dataset with 5,000 observations and three random variables, *z1*, *z2*, and *z3*, sampled from uncorrelated standard normal distributions. Options could specify other means, standard deviations, and correlation or covariance matrices.

- **dropmiss**

Automatically drops from the dataset in memory any *variables* that have missing values for every observation. This can be useful when working with a subset from a larger dataset, where some of the original variables are not applicable to any of the remaining observations. Typing **dropmiss, obs** will instead drop from memory any *observations* that have missing values for every variable. **dropmiss** is a user-written program not supplied directly with Stata. Type **findit dropmiss** for links to download and install it.

- **edit**

Opens the spreadsheet-like Data Editor where data can be entered or edited. Alternatively, use the Data menu or click 

- **edit year month extent**

Opens the Data Editor with only the variables *year*, *month* and *extent* (in that order) visible and available for editing.

- **encode stringvar, gen(numvar)**

Creates a new variable named *numvar*, with labeled numeric values based on the string (non-numeric) variable *stringvar*.

- **format rainfall %8.2f**

Establishes a fixed (f) display format for numeric variable *rainfall*: 8 columns wide, with two digits always shown after the decimal. This affects only how values are displayed.

- **generate newvar = (x + y)/100**

Creates a new variable named *newvar*, equal to *x* plus *y* divided by 100.

- **generate newvar = runiform()**

Creates a new variable with values sampled from a uniform random distribution over the interval ranging from 0 to nearly 1, written [0,1). Type **help random** to see functions for generating random data from normal, binomial, χ^2 , gamma, Poisson and other distributions.

- **import excel filename.xlsx, sheet("mean") cellrange(a15:n78) firstrow**

Imports an Excel spreadsheet into memory. Options in this example specify the sheet named “mean,” containing the data of interest in cells A15 through N78. The first row of this data area gives variable names.

- **infile x y z using data.raw**

Reads a text file named *data.raw* containing data on three variables: *x*, *y* and *z*. The values of these variables are separated by one or more white-space characters — blanks, tabs and newlines (carriage return, linefeed, or both) — or by commas. With white-space delimiters, missing values for numerical variables must be represented by periods, not blanks. With comma-delimited data, missing values are represented by a period or by two consecutive commas. Stata also provides for extended missing values, as discussed later in this chapter. Other commands are better suited for reading tab-delimited, comma-delimited or fixed-column raw data; type **help infiling** for more information.

- **list**

Lists the data in default or table format. With large datasets, table format becomes hard to read, and **list, display** produces better results. See **help list** for other options. The Data Editor or Data Browser provide more useful views for many purposes.

- **list x y z in 5/20**

Lists the *x*, *y* and *z* values of the 5th through 20th observations, as the data are presently sorted. The **in** qualifier works in similar fashion with most other Stata commands as well.

- **merge 1:1 id using olddata**

Reads the previously-saved dataset *olddata.dta* and matches observations from *olddata* 1-to-1 with observations in memory that have identical *id* values. Both *olddata* (the “using” data) and the data currently in memory (the “master” data) must already be sorted by *id*.

- **mvdecode var3-var62, mv(97=.\ \ 98=.a \ 99=.b)**

For variables *var3* through *var62*, recode the numerical values 97, 98 and 99 as missing. In this example we use three separate missing value codes, which Stata represents as a period, .a and .b. These could represent different reasons the values are missing, such as responses of “Not applicable,” “Don’t know” and “Refused to answer” on a survey. If only one missing-value code is required, we can instead specify an option such as

mv(97 98 99=.)

- . **replace oldvar = 100 * oldvar**
Replaces the values of *oldvar* with 100 times their previous values.
- . **sample 10**
Drops all the observations in memory except for a 10% random sample. Instead of selecting a certain percentage, we could select a certain number of cases. For example, **sample 55**, **count** would drop all but a random sample of size $n = 55$.
- . **save newfile**
Saves the data currently in memory, as a file named *newfile.dta*. If *newfile.dta* already exists, and you want to write over the previous version, type **save newfile, replace**. Alternatively, use the File menu. To save *newfile.dta* in the format of Stata version 9, type **saveold newfile** or select File > Save As > Save as type .
- . **sort x**
Sorts the data from lowest to highest values of *x*. Observations with missing *x* values appear last after sorting because Stata views missing values as very high numbers. Type **help gsort** for a more general sorting command that can arrange values in either ascending or descending order and can optionally place the missing values first.
- . **tabulate x if y > 65**
Produces a frequency table for *x* using only those observations that have *y* values above 65. The **if** qualifier works similarly with most other Stata commands.
- . **use oldfile**
Retrieves previously-saved Stata-format dataset *oldfile.dta* from disk, and places it in memory. If other data are currently in memory and you want to discard those data without saving them, type **use oldfile, clear**. Alternatively, these tasks can be accomplished through File > Open or by clicking 

Creating a New Dataset by Typing in Data

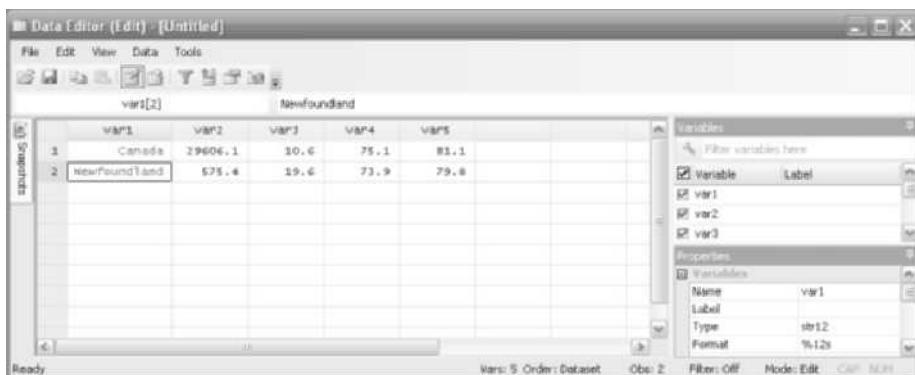
Data that were previously saved in Stata format can be retrieved into memory either by typing a command of the form **use filename**, or by menu selections. This section describes basic tricks for creating Stata-format datasets in the first place. We could start simply by typing data into the Data Editor by hand. A by-hand approach is practical with small datasets, or may be unavoidable when the original information is printed material such as a table in a book. If the original information is in electronic format such as a text file or spreadsheet, however, more direct approaches are possible.

Table 2.1 lists some information about Canadian provinces and territories that can be used to illustrate the by-hand approach. These data are from the Federal, Provincial and Territorial Advisory Committee on Population Health, 1996. Canada's newest territory, Nunavut, is not listed here because it was part of the Northwest Territories until 1999.

Table 2.1: Data on Canada and Its Provinces

Place	1995 Pop. (1000's)	Unemployment Rate (percent)	Male Life Expectancy (years)	Female Life Expectancy (years)
Canada	29606.1	10.6	75.1	81.1
Newfoundland	575.4	19.6	73.9	79.8
Prince Edward Island	136.1	19.1	74.8	81.3
Nova Scotia	937.8	13.9	74.2	80.4
New Brunswick	760.1	13.8	74.8	80.6
Quebec	7334.2	13.2	74.5	81.2
Ontario	11100.3	9.3	75.5	81.1
Manitoba	1137.5	8.5	75.0	80.8
Saskatchewan	1015.6	7.0	75.2	81.8
Alberta	2747.0	8.4	75.5	81.4
British Columbia	3766.0	9.8	75.8	81.4
Yukon	30.1	—	71.3	80.4
Northwest Territories	65.8	—	70.2	78.0

The simplest way to create a dataset from printed information like Table 2.1 is through the Data Editor, invoked by clicking , selecting Window > Data Editor from the menu bar, or by typing the command **edit**. Then begin typing values for each variable, in columns initially labeled *var1*, *var2* etc. Thus, *var1* contains place names, *var2* populations, and so forth.



We can assign more descriptive variable names by double-clicking on the column headings (such as *var1*) and then typing a new name in the resulting dialog box; eight characters or fewer works best, although names with up to 32 characters are allowed. We can also create variable labels that contain a brief description. For example, *var2* (population) might be renamed *pop*, and given the variable label “Population in 1000s, 1995”.

Renaming and labeling variables can also be done outside of the Data Editor through the **rename** and **label variable** commands:

```
. rename var2 pop
. label variable pop "Population in 1000s, 1995"
```

Cells left empty, such as unemployment rates for the Yukon and Northwest Territories, will automatically be assigned Stata's default missing value code, a period. At any time, we can close the Data Editor and then save the dataset to disk. Clicking or Data > Data Editor , or typing the command **edit**, brings the Editor back.

If the first value entered for a variable is a number, as with population, unemployment and life expectancy, then Stata assumes that this column is a numeric variable and it will thereafter permit only numbers as values. Numeric values can also begin with a plus or minus sign, include decimal points, or be expressed in scientific notation. For example, we could represent Canada's population as 2.96061e+7, which means 2.96061×10^7 or about 29.6 million people. Numbers *should not include any commas*, such as 29,606,100 (or using commas as a decimal separator). If we did happen to put commas within the first value typed in a column, Stata would interpret this as a string variable (next paragraph) rather than as a number.

If the first value entered for a variable includes non-numeric characters, as did place names above (or “1,000” with the comma), then Stata thereafter considers this column to be a string or text variable. String variable values can be almost any combination of letters, numbers, symbols or spaces up to 244 characters long. They can store names, quotations or other descriptive information. String variable values could be tabulated and counted, but not analyzed using means, correlations or most other statistics. In the Data Editor or Data Browser, string variable values appear in red, distinguishing them from numeric (black) or labeled numeric (blue) variables.

After typing in the information from Table 2.1 in this fashion, we close the Data Editor and save our data, perhaps with the name *Canada1.dta*:

```
. save Canada1
```

Stata automatically adds the extension .dta to any dataset name, unless we tell it to do otherwise. If we already had saved and named an earlier version of this file, it is possible to write over that with the newest version by typing

```
. save, replace
```

At this point, our new dataset looks like this:

```
. describe
```

```
Contains data from C:\data\Canada1.dta
obs: 13
vars: 5
size: 481
1 Jul 2012 17:42
```

variable name	storage type	display format	value label	variable label
var1	str21	%21s		
pop	float	%9.0g		Population in 1000s, 1995
var3	float	%9.0g		
var4	float	%9.0g		
var5	float	%9.0g		

Sorted by:

. list

		var1	pop	var3	var4	var5
1.	Prince Edward Island	Canada	29606.1	10.6	75.1	81.1
2.		Newfoundland	575.4	19.6	73.9	79.8
3.		Edward Island	136.1	19.1	74.8	81.3
4.		Nova Scotia	937.8	13.9	74.2	80.4
5.		New Brunswick	760.1	13.8	74.8	80.6
6.	Alberta	Quebec	7334.2	13.2	74.5	81.2
7.		Ontario	11100.3	9.3	75.5	81.1
8.		Manitoba	1137.5	8.5	75	80.8
9.		Saskatchewan	1015.6	7	75.2	81.8
10.		Alberta	2747	8.4	75.5	81.4
11.	Northwest Territories	British Columbia	3766	9.8	75.8	81.4
12.		Yukon	30.1	:	71.3	80.4
13.		Northwest Territories	65.8	:	70.2	78

. summarize

Variable	Obs	Mean	Std. Dev.	Min	Max
var1	0				
pop	13	4554.769	8214.304	30.1	29606.1
var3	11	12.10909	4.250048	7	19.6
var4	13	74.29231	1.673052	70.2	75.8
var5	13	80.71539	.9754027	78	81.8

Examining such output gives us a chance to look for errors that should be corrected. The **summarize** table, for instance, provides several numbers useful in proofreading, including the count of nonmissing numerical observations (always 0 for string variables) and the minimum and maximum for each variable. Substantive interpretation of the summary statistics would be premature at this point, because our dataset contains one observation (Canada) that represents a combination of the other 12 provinces and territories.

The next step is to make our dataset more self-documenting. The variables could be given more descriptive names, such as the following:

```
. rename var1 place
. rename var3 unemp
. rename var4 mlife
. rename var5 flife
```

Alternatively, the four **rename** operations could be accomplished in one step:

```
. rename (var1 var2 var3 var4) (place unemp mlife flife)
```

Stata also permits us to add several kinds of labels to the data. **label data** describes the dataset as a whole, whereas **label variable** describes an individual variable. For example,

```
. label data "Canadian dataset 1"
. label variable place "Place name"
. label variable unemp "% 15+ population unemployed, 1995"
. label variable mlife "Male life expectancy years"
. label variable flife "Female life expectancy years"
```

By labeling data and variables, we obtain a dataset that is more self-explanatory:

```
. describe

Contains data from C:\data\Canada1.dta
    obs:                 13                               Canadian dataset 1
    vars:                  5                               4 Jul 2012 11:21
    size:                481

variable   storage   display   value
name      type      format    label      variable label
place      str21    %21s      Place name
pop        float     %9.0g    Population in 1000s, 1995
unemp     float     %9.0g    % 15+ population unemployed, 1995
mlife      float     %9.0g    Male life expectancy years
flife      float     %9.0g    Female life expectancy years

Sorted by:
Note: dataset has changed since last saved
```

Once labeling is completed, we should save the data to disk by using File > Save or typing

```
. save, replace
```

We could later retrieve these data any time through  , File > Open, or by typing

```
. use C:\data\Canada1
```

Now we can proceed with analysis. We might notice, for instance, that male and female life expectancies correlate positively with each other and also negatively with the unemployment rate. The life expectancy–unemployment rate correlation is stronger for males.

```
. correlate unemp mlife flife
(obs=11)
```

	unemp	mlife	flife
unemp	1.0000		
mlife	-0.7440	1.0000	
flife	-0.6173	0.7631	1.0000

The order of observations within a dataset can be changed by the **sort** command. For example, to rearrange observations from smallest to largest in population, type

```
. sort pop
```

String variables are sorted alphabetically instead of numerically. Typing **sort place** will rearrange observations putting Alberta first, British Columbia second, and so on.

The **order** command controls the order of variables within a dataset. For example, we could make unemployment the second variable, and population last:

```
. order place unemp mlife flife pop
```

The Data Editor also offers a Tools menu with choices that can perform these operations.

We can restrict the Data Editor beforehand to work only with certain variables, in a specified order, or with a specified range of values. For example,

```
. edit place mlife flife
```

or

```
. edit place unemp if pop > 100
```

The last example employs an **if** qualifier, an important tool described in later sections.

Creating a New Dataset by Copy and Paste

When the original data source is electronic, such as a web page, text file, spreadsheet or word processor document, we can bring these data into Stata by copy and paste. For example, the National Climate Data Center (NCDC) produces estimates of global temperature anomalies (deviations from the 1901–2000 mean, in degrees Celsius) for every month back to January 1880. The NCDC index is one of several based on a global network of data from weather stations and sea surface measurements. NCDC updates the global index monthly (through December 2012 as this is written) and publishes results online. The first five months are listed below. The first value, -0.0623, indicates that January 1880 was globally about .06 °C cooler than the average for January in the 20th Century.

```
1880 1 -0.0623
1880 2 -0.1929
1880 3 -0.1966
1880 4 -0.0912
1880 5 -0.1510
```

Depending on details of how raw data (including missing values) are organized, it may not work to just copy the whole set of numbers and paste them into the Data Editor. An intermediate step, expressing the raw data as comma-separated values, often proves helpful. An easy way to do this is to copy all the numbers and paste them into Stata's Do-File Editor, a simple text editor that has many applications. Then use the Do-File Editor's Edit > Find > Replace function to Replace All occurrences of double spaces with single spaces. Repeat this a few times until no double spaces (only single spaces) remain in the document. Then as a last step, Replace All the

single spaces with commas. We have just used the Do-File Editor to convert the data into comma separated values, a very common data format. In the Do-File Editor, we can also add a first row containing comma-separated variable names:

```
year,month,temp
1880,1,-0.0623
1880,2,-0.1929
1880,3,-0.1966
1880,4,-0.0912
1880,5,-0.1510
```

We can now Edit > Select All then copy the information from the Do-File Editor and paste it into an empty Data Editor, using Paste Special with Comma delimiter and Treat first row as variable names options.



Comma-separated values (.csv) files can also be written by any spreadsheet, or by Stata itself, making this a conveniently portable data format. To read a .csv file directly into Stata use an **insheet** command:

```
. insheet using C:\data\global.csv, comma clear
```

Once data are in memory, we can label the data and variables, then save the results as a Stata system file.

```
. label data "Global climate"
. label variable year "Year"
. label variable month "Month"
. label variable temp "NCDC global temp anomaly vs 1901-2000, C"
. save C:\data\global1.dta
. describe
```

```

Contains data from C:\data\global1.dta
obs:      1,584                               Global climate
vars:      3                                    12 Feb 2012 08:50
size:     11,088

variable   storage   display   value
name      type       format    label
year       int        %8.0g    Year
month      byte       %8.0g    Month
temp       float      %9.0g   NCDC global temp anomaly vs
                           1901-2000, C

```

Sorted by:

Specifying Subsets of the Data: in and if Qualifiers

Many Stata commands can be restricted to a subset of the data by adding an **in** or **if** qualifier. Qualifiers are also available for many menu selections: look for an **if/in** or **by/if/in** tab along the top of the dialog. **in** specifies the observation numbers to which the command applies. For example, **list in 5** tells Stata to list only the 5th observation. To list the 1st through 5th observations, type

```
. list in 1/5
```

	year	month	temp
1.	1880	1	-.0623
2.	1880	2	-.1929
3.	1880	3	-.1966
4.	1880	4	-.0912
5.	1880	5	-.151

The letter **I** denotes the last case, and **-10**, for example, the tenth-from-last. Among the 1,584 months in our global temperature data, which 10 months had the highest temperature anomalies, meaning they were farthest above the 1901–2000 average for that month? To find out, we first sort from lowest to highest by temperature, then list the 10th-from-last to last observations:

```
. sort temp
. list in -10/1
```

	year	month	temp
1575.	1998	4	.7241
1576.	2003	12	.7317
1577.	2004	11	.7399
1578.	2006	12	.7417
1579.	2010	4	.7515
1580.	2002	3	.7704
1581.	2002	2	.7784
1582.	2010	3	.7802
1583.	1998	2	.8388
1584.	2007	1	.8422

Note the important, although typographically subtle, distinction between **1** (number one, or first observation) and **I** (letter “el,” or last observation). The **in** qualifier works in a similar way with most other analytical or data-editing commands. It always refers to the data *as presently sorted*.

The **if** qualifier also has broad applications, but it selects observations based on specific variable values. For example, to see the mean and standard deviation of temperature anomalies prior to 1970, type

```
. summarize temp if year < 1970
```

Variable	Obs	Mean	Std. Dev.	Min	Max
temp	1080	-.1232613	.1829313	-.7316	.4643

To summarize temperatures in more recent years, type

```
. summarize temp if year >= 1970
```

Variable	Obs	Mean	Std. Dev.	Min	Max
temp	504	.3159532	.2300395	-.2586	.8422

The “<” (is less than) and “>=” (greater than or equal to) signs are *relational operators*:

- ==** is equal to
- !=** is not equal to (**~=** also works)
- >** is greater than
- <** is less than
- >=** is greater than or equal to
- <=** is less than or equal to

A double equals sign, “**==**”, denotes the logical test, “Is the value on the left side the same as the value on the right?” To Stata, a single equals sign means something different: “*Make* the value on the left side be the same as the value on the right.” The single equals sign is not a relational operator and cannot be used within **if** qualifiers. Single equals signs have other meanings. They are used with commands that generate new variables, or replace the values of old ones, according to algebraic expressions. Single equals signs also appear in certain specialized applications such as weighting and hypothesis tests.

Two or more relational operators can be combined within a single **if** expression by the use of *logical operators*. Stata’s logical operators are the following:

- &** and
- |** or (symbol is a vertical bar, not the number one or letter “el”)
- !** not (**~** also works)

Parentheses allow us to specify the precedence among multiple operators. The following command will summarize January and February temperature anomalies for the years from 1940 through 1969:

```
. summarize temp if (month == 1 | month == 2) & year >= 1940  
& year < 1970
```

A note of caution regarding missing values: Stata ordinarily shows missing values as a period, but in some operations (notably **sort** and **if**, although not in statistical calculations such as means

or correlations), these same missing values are treated as if they were large positive numbers. For example, suppose that we are analyzing opinion poll data. A command such as the following would tabulate *vote* not only for people age 65 and older, as intended, but also for any people whose *age* values are missing:

```
. tabulate vote if age >= 65
```

Where missing values exist, we often need to deal with them explicitly in the **if** expression.

```
. tabulate vote if age >= 65 & !missing(age)
```

The not missing() function **!missing()** provides a general way to select observations with nonmissing values. As shown later in this chapter, Stata permits up to 27 different missing values codes, although so far we have used only the default “.”. **if !missing(*age*)** sets them all aside. Type **help missing** for more details.

There are several alternative ways to screen out missing values. The **missing()** function evaluates to 1 if a value is missing, and 0 if it is not. For example, to tabulate *vote* only for those observations that have nonmissing values of *age*, *income* and *education*, type

```
. tabulate vote if missing(age, income, education)==0
```

Finally, because the default missing value “.” is represented internally by a very large number, and other missing values (described later) are even larger, a “less than” inequality **<** can be used to screen all of them out:

```
. tabulate vote if age <. & income <. & education <.
```

The **in** and **if** qualifiers set observations aside temporarily so that a particular command does not apply to them. These qualifiers have no effect on the data in memory, and the next command will apply to all observations unless it too has an **in** or **if** qualifier. To drop variables from the data in memory, use the **drop** command (or use the Data Editor). Returning to our Canadian data (*Canada1.dta*), we could drop *mlife* and *flife* from memory by typing

```
. drop mlife flife
```

Either **in** or **if** qualifiers can be used to select which observations to drop. For example, **drop in 12/13** means to drop the 12th and 13th observation in a dataset. We can also drop selected variables or observations with the Delete button in the Data Editor.

Instead of telling Stata which variables or observations to drop, it sometimes is simpler to specify which to keep. Rather than **drop mlife** and **flife** from the *Canada1.dta* data, we accomplish the same thing if we **keep** the other three variables.

```
. keep place pop unemp
```

Like any other changes to the data in memory, none of these reductions affect disk files until we save the data. At that point, we will have the option of writing over the old dataset (**save**,

replace) and thus destroying it, or just saving the newly modified dataset with a new name (by choosing File > Save As , or by typing a command with the form **save *newname***) so that both versions exist on disk.

Generating and Replacing Variables

The **generate** and **replace** commands allow us to create new variables or change the values of existing variables. For example, in Canada, as in most industrial societies, women tend to live longer than men. To analyze regional variations in this gender gap, we might retrieve dataset *Canada1.dta* and generate a new variable equal to female life expectancy (*flife*) minus male life expectancy (*mlife*). In the main part of a **generate** or **replace** statement (unlike **if** qualifiers) we use a single equals sign.

```
. use C:\data\Canada1, clear
. generate gap = flife - mlife
. label variable gap "Female-male life expectancy gap"
. describe gap

variable   storage   display      value
name      type      format     label      variable label
gap        float    %9.0g          Female-male life expectancy gap

. list place flife mlife gap



|     | place                 | flife | mlife | gap      |
|-----|-----------------------|-------|-------|----------|
| 1.  | Canada                | 81.1  | 75.1  | 6        |
| 2.  | Newfoundland          | 79.8  | 73.9  | 5.900002 |
| 3.  | Prince Edward Island  | 81.3  | 74.8  | 6.5      |
| 4.  | Nova Scotia           | 80.4  | 74.2  | 6.200005 |
| 5.  | New Brunswick         | 80.6  | 74.8  | 5.799995 |
| 6.  | Quebec                | 81.2  | 74.5  | 6.699997 |
| 7.  | Ontario               | 81.1  | 75.5  | 5.599998 |
| 8.  | Manitoba              | 80.8  | 75    | 5.800003 |
| 9.  | Saskatchewan          | 81.8  | 75.2  | 6.600006 |
| 10. | Alberta               | 81.4  | 75.5  | 5.900002 |
| 11. | British Columbia      | 81.4  | 75.8  | 5.599998 |
| 12. | Yukon                 | 80.4  | 71.3  | 9.099998 |
| 13. | Northwest Territories | 78    | 70.2  | 7.800003 |


```

For the province of Newfoundland, the true value of *gap* should be $79.8 - 73.9 = 5.9$ years, but the output shows this value as 5.900002 instead. Like all computer programs, Stata stores numbers in binary form, and 5.9 has no exact binary representation. The small inaccuracies that arise from approximating decimal fractions in binary are unlikely to affect statistical calculations much, but they appear disconcerting in data lists. We can change the display format so that Stata shows only a rounded-off version. The following command specifies a fixed display format four numerals wide, with one digit to the right of the decimal:

```
. format gap %4.1f
```

Even when the display shows 5.9, however, a command such as the following will return no observations:

```
. list if gap == 5.9
```

This occurs because Stata believes the value does not exactly equal 5.9. (More technically, Stata stores *gap* values in single, float precision but does all calculations in double precision, and the single- and double-precision approximations of 5.9 are not identical.)

Display formats, as well as variables names and labels, can also be changed by double-clicking on a column in the Data Editor. Fixed numeric formats such as **%4.1f** are one of the three most common numeric display format types. These are

- %w.dg** General numeric format, where *w* specifies the total width or number of columns displayed and *d* the minimum number of digits that must follow the decimal point. Exponential notation (such as 1.00e+07, meaning 1.00×10^7 or 10 million) and shifts in the decimal-point position will be used automatically as needed, to display values in an optimal (but varying) fashion.
- %w.df** Fixed numeric format, where *w* specifies the total width or number of columns displayed and *d* the fixed number of digits that must follow the decimal point.
- %w.de** Exponential numeric format, where *w* specifies the total width or number of columns displayed and *d* the fixed number of digits that must follow the decimal point.

For example, as we saw in Table 2.1, the 1995 population of Canada was approximately 29,606,100 people, and the Yukon Territory population was 30,100. The table below shows how those two numbers appear under several different display formats.

format	Canada	Yukon
%9.0g	2.96e+07	30100
%9.1f	29606100.0	30100.0
%12.5e	2.96061e+07	3.01000e+04

Although the displayed values look different, their internal values are identical. Calculations remain unaffected by display formats. Other numeric display formatting options include the use of commas, left- and right-justification, or leading zeros. There also exist special formats for dates, time series variables and string variables. Type **help format** for more information.

replace can make the same sorts of calculations as **generate**, but it changes values of an existing variable instead of creating a new variable. For example, suppose that we had data on income in dollars. We decide it would be more convenient to work with income in thousands of dollars. To convert dollars to thousands of dollars, we divide all values by 1,000:

```
. replace income = income/1000
```

replace can make such wholesale changes, or it can be used with **in** or **if** qualifiers to selectively edit the data. Suppose our survey variables include *age* and year *born*. A command such as the following would correct one or more typos where a subject's age had been incorrectly typed as 299 instead of 29:

```
. replace age = 29 if age == 299
```

Alternatively, the following command could correct an error in the value of *age* for observation number 1453:

```
. replace age = 29 in 1453
```

For a more complicated example,

```
. replace age = 2012-born if missing(age) | age+1 < 2012-born
```

This replaces values of variable *age* with 2012 minus the year of birth (*born*) if *age* is missing or if the reported age (plus one year) is less than 2012 minus the year of birth.

generate and **replace** provide tools to create categorical variables as well. We noted earlier that our Canadian dataset includes several types of observations: 2 territories, 10 provinces and one country combining them all. Although **in** and **if** qualifiers allow us to separate these, and **drop** can eliminate observations from the data, it might be most convenient to have a categorical variable that indicates the observation's type. The following example shows one way to create such a variable, using our *Canada1.dta* dataset. We start by generating *type* as a constant, equal to 1 for each observation. Next, we replace this with the value 2 for the Yukon and Northwest Territories, and with 3 for Canada. The final steps involve labeling new variable *type* and defining labels for values 1, 2 and 3.

```
. use C:\data\Canada1, clear
. generate type = 1
. replace type = 2 if place == "Yukon" | place == "Northwest
Territories"
. replace type = 3 if place == "Canada"
. label variable type "Province, territory or nation"
. label define typelbl 1 "Province" 2 "Territory" 3 "Nation"
. label values type typelbl
. list
```

	place	pop	unemp	mlife	flife	type
1.	Canada	29606.1	10.6	75.1	81.1	Nation
2.	Newfoundland	575.4	19.6	73.9	79.8	Province
3.	Prince Edward Island	136.1	19.1	74.8	81.3	Province
4.	Nova Scotia	937.8	13.9	74.2	80.4	Province
5.	New Brunswick	760.1	13.8	74.8	80.6	Province

6.	Quebec	7334.2	13.2	74.5	81.2	Province
7.	Ontario	11100.3	9.3	75.5	81.1	Province
8.	Manitoba	1137.5	8.5	75	80.8	Province
9.	Saskatchewan	1015.6	7	75.2	81.8	Province
10.	Alberta	2747	8.4	75.5	81.4	Province
11.	British Columbia	3766	9.8	75.8	81.4	Province
12.	Yukon	30.1	.	71.3	80.4	Territory
13.	Northwest Territories	65.8	.	70.2	78	Territory

As illustrated, labeling the values of a categorical variable requires two commands. The **label define** command specifies what labels go with what numbers. The **label values** command specifies to which variable these labels apply. One set of labels (created through one **label define** command) can apply to any number of variables (that is, be referenced in any number of **label values** commands). Value labels can have up to 32,000 characters, but work best for most purposes if they are not too long.

generate can create new variables, and **replace** can produce new values, using any mixture of old variables, constants, random values and expressions. For numeric variables, the following *arithmetic operators* apply:

- + add
- subtract
- * multiply
- / divide
- ^ raise to power

Parentheses will control the order of calculation. Without them, the ordinary rules of precedence apply. Of the arithmetic operators, only addition, “+”, works with string variables, where it connects two string values into one.

Although their purposes differ, **generate** and **replace** have similar syntax. Either can use any mathematically or logically feasible combination of Stata operators and **in** or **if** qualifiers. These commands can also employ Stata’s broad array of special functions, introduced later.

Missing Value Codes

Examples seen so far involve only a single missing-value code, Stata’s default: a large number which Stata displays as a period. In some datasets, however, values might be missing for several different reasons. We could denote different kinds of missing values by using extended missing-value codes. These are even larger numbers, which Stata displays as “.a” through “.z”. Unlike the default missing-value code “.”, the extended missing-value codes can be labeled.

Different kinds of missing values often arise with surveys, where the question “In what year were you married?” might have no answer because the respondent has never been married, can’t recall, or thinks it’s none of your business. Dataset *Granite2011_6.dta*, contains data from a political opinion survey, New Hampshire’s Granite State Poll. A question asking respondents about their level of interest regarding the 2012 general election (*genint*) serves to illustrate Stata’s extended missing-value codes.

At first glance, *genint* appears straightforward, but for many analyses this variable would be awkward to use.

. **tabulate genint**

Interest in 2012 pres. election	Freq.	Percent	Cum.
extremely interested	245	47.48	47.48
very interested	168	32.56	80.04
somewhat interested	72	13.95	93.99
not very interested	28	5.43	99.42
don't know	2	0.39	99.81
no answer	1	0.19	100.00
Total	516	100.00	

The first four values, labeled “extremely interested” to “not very interested” form a scale of disinterest. The last two, “don’t know” and “no answer” are not part of this scale but two different kinds of non-answers. Like many other surveys, the Granite State Poll employs particular numbers to represent various non-answers. In this case, the number 98 means the respondent said he or she did not know how interested they were, and 99 means no answer was given. We can see these numerical values if we ask for the same table without value labels.

. **tabulate genint, nolabel**

Interest in 2012 pres. election	Freq.	Percent	Cum.
1	245	47.48	47.48
2	168	32.56	80.04
3	72	13.95	93.99
4	28	5.43	99.42
98	2	0.39	99.81
99	1	0.19	100.00
Total	516	100.00	

Any statistics calculated for *genint* will be confused by the 98 and 99 codes. For example, a table of *genint* means by respondent education will be meaningless, because those 98 and 99 values have been averaged in.

. **tabulate educ, summarize(genint)**

Highest degree completed	Summary of Interest in 2012 pres. election		
	Mean	Std. Dev.	Freq.
HS or less	2.8275862	8.9668722	116
Tech/some	3.5	12.451587	120
College grad	1.672956	.82290667	159
Postgrad	1.5775862	.80380467	116
Total	2.3424658	7.4366697	511

We need an improved version of this variable, to be called *genint2*. This new version will be different in three ways. First, we reverse the 1 through 4 values so that higher values indicate greater interest instead of less interest — making interpretation more natural.

```
. generate genint2 = 5 - genint if genint < 90
```

Second, the 98 and 99 values should be identified as missing so they do not enter calculations for the mean and other statistics. Here we use the missing value code .a to represent “don’t know” responses, formerly coded 98. We use .b to represent “no answer” responses, formerly coded 99.

```
. replace genint2 = .a if genint == 98
. replace genint2 = .b if genint == 99
```

Third, the value labels can be shortened from long phrases like “extremely interested” to something that will take up less space in graphs and tables.

```
. label variable genint2 "Interest in 2012 election (new)"
. label define genint2 1 "Not very" 2 "Somewhat" 3 "Very"
    4 "Extremely" .a "DK" .b "NA"
. label values genint2 genint2
```

Finally, a very important step: tabulating old against new variables to be sure that our commands worked as intended.

```
. tabulate genint genint2, miss
```

Interest in 2012 pres. election	Interest in 2012 election (new)				Total
	Not very	Somewhat	Very	Extremely	
extremely interested	0	0	0	245	245
very interested	0	0	168	0	168
somewhat interested	0	72	0	0	72
not very interested	28	0	0	0	28
don't know	0	0	0	0	2
no answer	0	0	0	0	1
Total	28	72	168	245	516

Interest in 2012 pres. election	Interest in 2012 election (new)		Total
	DK	NA	
extremely interested	0	0	245
very interested	0	0	168
somewhat interested	0	0	72
not very interested	0	0	28
don't know	2	0	2
no answer	0	1	1
Total	2	1	516

With these changes we have a more analyzable version. For example, it is easy to see that the average level of interest in the election rises with education.

```
. tabulate educ, summ(genint2)
```

Highest degree completed	Summary of interest in 2012 election (new)		
	Mean	Std. Dev.	Freq.
HS or less	3	.98229949	115
Tech/some College	3.1101695	.89427331	118
Postgrad	3.327044	.82290667	159
Total	3.4224138	.80380467	116
	3.2244094	.88647221	508

Any time we encounter specific numbers (such as 98 and 99 in the example above) used to indicate missing values, it is advisable to change these to missing value codes so that Stata does not enter the fake numbers into statistical calculations. This could be done easily for a whole list of variables using an **mvdecode** command such as

```
. mvdecode genint income age, mv(97=.) \ 98=.a \ 99=.b
```

The example above would change any values of *genint*, *income* or *age* from 97 to “.”, from 98 to “.a” and so forth. The .a and .b (through .z) missing values can accept value labels, but “.” by itself cannot.

As usual, the changes we have made do not become permanent until our dataset is saved. After so much recoding, it makes sense to save these data with a new name — in case, for some future reason, we want to take another look at the original raw data.

Using Functions

This section lists many of the functions available for use with **generate** or **replace**. For example, we could create a new variable named *loginc*, equal to the natural logarithm of *income*, by using the natural log function **ln** in a **generate** command:

```
. generate loginc = ln(income)
```

ln is one of Stata's *mathematical functions*. Other examples include **log10(x)** for base 10 logarithms; **int(x)** for the integer portion of *x*; **exp(x)** for the exponential (*e* to power) of *x*. There are many others; see **help math functions** for a complete list with details.

Many *probability density functions* exist as well. Consult **help density functions** and the reference manuals for a full list and details such as definitions, constraints on parameters, and the treatment of missing values. For example, **invnormal(p)** gives the inverse cumulative standard normal distribution, or the *z* value corresponding to probability *p*. Other functions include beta, binomial, chi-squared, *t*, *F*, gamma and uniform distributions. Of particular interest for simulation purposes, **runiform()** uses a pseudo-random number generator to return values from a uniform distribution theoretically ranging from 0 to nearly 1, written [0,1).

Stata provides many *date functions*, date-related *time series functions*, and special formats for displaying time or date variables. Lists and details can be found in the *User's Guide*, or by

typing **help date functions**. Date functions often involve elapsed dates, which refer to the number of days since January 1, 1960.

The global temperature dataset we built earlier in this chapter provides an example for elapsed dates. The file contains *year* and *month*, but no variable that combines both into a single measure of time.

```
. use C:\data\global1.dta, clear
. describe

Contains data from C:\data\global1.dta
obs:      1,584                               Global climate
vars:       3                                  12 Feb 2012 08:50
size:    11,088

variable   storage  display     value
name      type    format    label    variable label
year        int     %8.0g      Year
month      byte    %8.0g      Month
temp       float   %9.0g    NCDC global temp anomaly vs
                           1901-2000, C

Sorted by:
```

We can generate a new elapsed-date variable, *edate*, by using the **mdy** (month, day, year) function. The global temperature data are monthly averages, so for “day” we might just use the 15th of each month. (For an alternative approach using monthly data, see the discussion of dataset *Climate.dta* in Chapter 12.) Because *edate* represents the number of days since January 1, 1960, dates before 1960 appear as negative numbers.

```
. generate edate = mdy(month, 15, year)
. label variable edate "elapsed date"
. list in 1/5
```

	year	month	temp	edate
1.	1880	1	-.0623	-29205
2.	1880	2	-.1929	-29174
3.	1880	3	-.1966	-29145
4.	1880	4	-.0912	-29114
5.	1880	5	-.151	-29084

A more readable dataset results if we format *edate* as a date variable (%**tdmCY**) showing month (**m**), century (**C**) and year (**Y**). Then the numerical *edate* -29205 takes the label “Jan1880”.

```
. format edate %tdmCY
. list in 1/5
```

	year	month	temp	edate
1.	1880	1	-.0623	Jan1880
2.	1880	2	-.1929	Feb1880
3.	1880	3	-.1966	Mar1880
4.	1880	4	-.0912	Apr1880
5.	1880	5	-.151	May1880

Finally, we **save** our data with the new variable. By graphing the global temperature anomaly *temp* against *edate*, we can draw a basic time plot.

```
. sort year month
. order year month edate
. save c:\data\global2.dta, replace
. graph twoway line temp edate
```

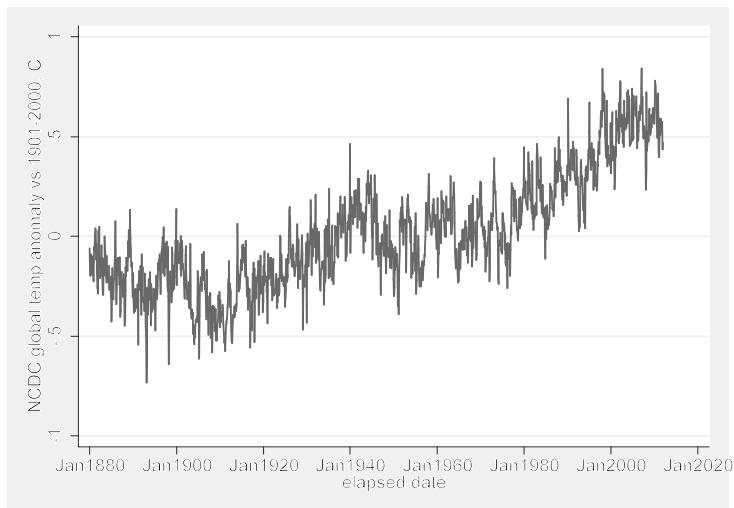


Figure 2.1

Other types of functions include *matrix functions*, *random number functions*, *string functions*, *time series functions* and *programming functions*. Type **help** followed by any of these terms to see a complete list. The reference manuals and *User's Guide* give further examples and details.

Multiple functions, operators and qualifiers can be combined in one command as needed. The functions and algebraic operators just described can also be used in another way that does not create or change any dataset variables. The **display** command performs a single calculation and shows the results onscreen. For example:

```
. display 2+3
5
. display log10(10^83)
83
. display invttail(120,.025) * 34.1/sqrt(975)
2.1622305
```

Thus, **display** can serve as an onscreen statistical calculator.

Unlike a calculator, **display**, **generate** and **replace** have direct access to Stata's statistical results. For illustration we return to the Arctic sea ice data introduced in Chapter 1, *Arctic9.dta*. One variable, *extent*, represents the mean area covered by at least 15% sea ice in September each

year (graphed earlier in Figure 1.1). For these 33 years of satellite observation, the overall September mean was about 6.52 million km².

```
. summarize extent
```

Variable	Obs	Mean	Std. Dev.	Min	Max
extent	33	6.51697	.9691796	4.3	7.88

After **summarize**, Stata temporarily stores this mean as a scalar named **r(mean)**.

```
. display r(mean)
6.5169697
```

We could use this result to create variable *extent0*, defined as the anomaly or deviation from the 1979–2011 mean. *extent0* will have the same standard deviation as *extent*, but a mean of approximately zero. It reflects how far above or below average each September value is.

```
. gen exten0 = extent - r(mean)
. summ extent extent0
```

Variable	Obs	Mean	Std. Dev.	Min	Max
extent	33	6.51697	.9691796	4.3	7.88
extent0	33	1.17e-08	.9691796	-2.216969	1.36303

Stata temporarily saves results after many analyses, such as **r(mean)** after **summarize**. These can be valuable for subsequent calculations or programming. To see a complete list of the names and values currently saved, type **return list**. In this example, saved values named **r(N)**, **r(sum_w)**, **r(mean)**, and so forth describe the most recent **summarize** results for *extent*.

```
. return list
scalars:
r(N) = 33
r(sum_w) = 33
r(mean) = 1.17403088194e-08
r(var) = .9393091848549505
r(sd) = .9691796452954171
r(min) = -2.21696949005127
r(max) = 1.363030433654785
r(sum) = 3.87430191040e-07
```

Stata also provides another variable-creation command, **egen** (extensions to **generate**), which has its own set of functions to accomplish tasks not easily done by **generate**. These include such things as creating new variables from the sums, maxima, minima, medians, interquartile ranges, standardized values, ranks or moving averages of existing variables or expressions. For example, the following command creates a new variable named *zscore*, equal to the standardized (mean 0, variance 1) values of *x*:

```
. egen zscore = std(x)
```

Or, the following command creates new variable *avg*, equal to the row mean of each observation's values on *x*, *y*, *z* and *w*, ignoring any missing values.

```
. egen avg = rowmean(x,y,z,w)
```

To create a new variable named *total*, equal to the row sum of each observation's values on *x*, *y*, *z*, and *w*, treating missing values as zeros, type

```
. egen total = rowtotal(x,y,z,w)
```

The following command creates new variable *xrank*, holding ranks corresponding to values of *x*: *xrank* = 1 for the observation with highest *x*. *xrank* = 2 for the second highest, and so forth.

```
. egen xrank = rank(x)
```

Consult **help egen** for a complete list of **egen** functions, or the reference manuals for further examples.

Converting Between Numeric and String Formats

Dataset *Canada2.dta* contains one string variable, *place*. It also has a labeled categorical variable, *type*. Both seem to have nonnumeric values.

```
. use C:\data\Canada2, clear
. list place type
```

	place	type
1.	Canada	Nation
2.	Newfoundland	Province
3.	Prince Edward Island	Province
4.	Nova Scotia	Province
5.	New Brunswick	Province
6.	Quebec	Province
7.	Ontario	Province
8.	Manitoba	Province
9.	Saskatchewan	Province
10.	Alberta	Province
11.	British Columbia	Province
12.	Yukon	Territory
13.	Northwest Territories	Territory

Beneath the labels, however, *type* remains a numeric variable, indicated by a blue font in the Data Editor or Brower. Clicking on that cell will show the underlying numbers, or we can **list** these asking for the **nolabel** option:

```
. list place type, nolabel
```

	place	type
1.	Canada	3
2.	Newfoundland	1
3.	Prince Edward Island	1
4.	Nova Scotia	1
5.	New Brunswick	1

6.	Quebec	1
7.	Ontario	1
8.	Manitoba	1
9.	Saskatchewan	1
10.	Alberta	1
11.	British Columbia	1
12.	Yukon	2
13.	Northwest Territories	2

String and labeled numeric variables behave differently when analyzed. Most statistical operations and algebraic relations are not defined for string variables, so we might want to have both string and labeled-numeric versions of the same information in our data. The **encode** command generates a labeled-numeric variable from a string variable. The number 1 is given to the alphabetically first value of the string variable, 2 to the second, and so on. The following example creates a labeled numeric variable named *placenum* from the string variable *place*:

```
. encode place, gen(placenum)
```

An opposite conversion is possible, too: The **decode** command generates a string variable using the values of a labeled numeric variable. Here we create string variable *typestr* from numeric variable *type*:

```
. decode type, gen(typestr)
```

When listed, the new numeric variable *placenum*, and the new string variable *typestr*, look similar to the originals:

```
. list place placenum type typestr
```

	place	placenum	type	typestr
1.	Canada	Canada	Nation	Nation
2.	Newfoundland	Newfoundland	Province	Province
3.	Prince Edward Island	Prince Edward Island	Province	Province
4.	Nova Scotia	Nova Scotia	Province	Province
5.	New Brunswick	New Brunswick	Province	Province
6.	Quebec	Quebec	Province	Province
7.	Ontario	Ontario	Province	Province
8.	Manitoba	Manitoba	Province	Province
9.	Saskatchewan	Saskatchewan	Province	Province
10.	Alberta	Alberta	Province	Province
11.	British Columbia	British Columbia	Province	Province
12.	Yukon	Yukon	Territory	Territory
13.	Northwest Territories	Northwest Territories	Territory	Territory

But with the **nolabel** option, the differences become visible. Stata views *placenum* and *type* basically as numbers.

```
. list place placenum type typestr, nolabel
```

		place	placenum	type	typestr
1.	Prince Edward Island	Canada	3	3	Nation
2.		Newfoundland	6	1	Province
3.		Prince Edward Island	10	1	Province
4.		Nova Scotia	8	1	Province
5.		New Brunswick	5	1	Province
6.	Alberta	Quebec	11	1	Province
7.		Ontario	9	1	Province
8.		Manitoba	4	1	Province
9.		Saskatchewan	12	1	Province
10.		Alberta	1	1	Province
11.	Northwest Territories	British Columbia	2	1	Province
12.		Yukon	13	2	Territory
13.		Northwest Territories	7	2	Territory

Most statistical analyses, such as finding means and standard deviations, work only with numeric variables. For calculation purposes, their labels do not matter.

```
. summarize place placenum type typestr
```

Variable	Obs	Mean	Std. Dev.	Min	Max
place	0				
placenum	13	7	3.89444	1	13
type	13	1.307692	.6304252	1	3
typestr	0				

Occasionally we encounter a string variable where the values are all or mostly numbers. To convert these string values into their numeric counterparts, use the **real** function. For example, in the artificial dataset below, the variable *siblings* is a string variable, although it only has one value, “4 or more,” that could not be represented just as well by a number.

```
. describe siblings
```

variable	name	storage type	display format	value label	variable label
	siblings	str9	%9s		Number of siblings (string)

```
. list
```

	siblings
1.	1
2.	3
3.	0
4.	2
5.	4 or more

```
. generate sibnum = real(siblings)
```

The new variable *sibnum* is numeric, with a missing value where *siblings* had “4 or more.”

```
. list
```

	siblings	sibnum
1.	1	1
2.	3	3
3.	0	0
4.	2	2
5.	4 or more	.

The **destring** command provides a more flexible method for converting string variables to numeric. In the example above, we could have accomplished the same thing by typing

```
. destring siblings, generate(sibnum) force
```

See **help destring** for information about syntax and options.

Creating New Categorical and Ordinal Variables

A previous section illustrated how to construct a categorical variable called *type* to distinguish among territories, provinces and nation in our Canadian dataset. You can create categorical or ordinal variables in many other ways. This section gives a few examples.

type has three categories:

```
. tabulate type
```

Province, territory or nation	Freq.	Percent	Cum.
Province	10	76.92	76.92
Territory	2	15.38	92.31
Nation	1	7.69	100.00
Total	13	100.00	

Suppose we want to re-express *type* as a set of dichotomies or dummy variables, each coded 0 or 1. **tabulate** will create dummy variables automatically if we add the **generate** option. In the following example, this results in a set of variables called *type1*, *type2* and *type3*, each representing one of the three categories of *type*:

```
. tabulate type, generate(type)
```

Province, territory or nation	Freq.	Percent	Cum.
Province	10	76.92	76.92
Territory	2	15.38	92.31
Nation	1	7.69	100.00
Total	13	100.00	

```
. describe
```

Contains data from c:\data\canada2.dta				Canadian dataset 2	
obs:	13	vars:	9	size:	572
variable name	storage type	display format	value label	variable label	
place	str21	%21s		Place name	
pop	float	%9.0g		Population in 1000s, 1995	
unemp	float	%9.0g		% 15+ population unemployed, 1995	
mlife	float	%9.0g		Male life expectancy years	
flife	float	%9.0g		Female life expectancy years	
type	float	%9.0g	type1b1	Province, territory or nation	
type1	byte	%8.0g		type==Province	
type2	byte	%8.0g		type==Territory	
type3	byte	%8.0g		type==Nation	

Sorted by:
Note: dataset has changed since last saved

```
. list place type type1-type3
```

	place	type	type1	type2	type3
1.	Canada	Nation	0	0	1
2.	Newfoundland	Province	1	0	0
3.	Prince Edward Island	Province	1	0	0
4.	Nova Scotia	Province	1	0	0
5.	New Brunswick	Province	1	0	0
6.	Quebec	Province	1	0	0
7.	Ontario	Province	1	0	0
8.	Manitoba	Province	1	0	0
9.	Saskatchewan	Province	1	0	0
10.	Alberta	Province	1	0	0
11.	British Columbia	Province	1	0	0
12.	Yukon	Territory	0	1	0
13.	Northwest Territories	Territory	0	1	0

Re-expressing categorical information as a set of dummy variables involves no loss of information; in this example, *type1* through *type3* together tell us exactly as much as *type* itself does. Occasionally, however, analysts choose to re-express a measurement variable in categorical or ordinal form, even though this *does* result in a substantial loss of information. For example, *unemp* in *Canada2.dta* gives a measure of the unemployment rate. Excluding Canada itself from the data, we see that *unemp* ranges from 7% to 19.6%, with a mean of 12.26:

```
. summarize unemp if type != 3
```

Variable	Obs	Mean	Std. Dev.	Min	Max
unemp	10	12.26	4.44877	7	19.6

Having Canada in the data becomes a nuisance at this point, so we drop it:

```
. drop if type == 3
```

Two commands create a dummy variable named *unemp2* with values of 0 when unemployment is below average (12.26), 1 when unemployment is equal to or above average, and missing when

unemp is missing. In reading the second command, recall that Stata's sorting and relational operators treat missing values as very large numbers.

```
. generate unemp2 = 0 if unemp < 12.26
(7 missing values generated)

. replace unemp2 = 1 if unemp >= 12.26 & !missing(unemp)
(5 real changes made)
```

We might want to group the values of a measurement variable, thereby creating an ordered-category or ordinal variable. The **autocode** function (see Using Functions) provides automatic grouping of measurement variables. To create new ordinal variable *unemp3*, which groups values of *unemp* into three equal-width groups over the interval from 5 to 20, type

```
. generate unemp3 = autocode(unemp, 3, 5, 20)
(2 missing values generated)
```

A list of the data shows how the new dummy (*unemp2*) and ordinal (*unemp3*) variables correspond to values of the original measurement variable *unemp*.

```
. list place unemp unemp2 unemp3
```

	place	unemp	unemp2	unemp3
1.	Newfoundland	19.6	1	20
2.	Prince Edward Island	19.1	1	20
3.	Nova Scotia	13.9	1	15
4.	New Brunswick	13.8	1	15
5.	Quebec	13.2	1	15
6.	Ontario	9.3	0	10
7.	Manitoba	8.5	0	10
8.	Saskatchewan	7	0	10
9.	Alberta	8.4	0	10
10.	British Columbia	9.8	0	10
11.	Yukon	:	:	:
12.	Northwest Territories	:	:	:

Using Explicit Subscripts with Variables

When Stata has data in memory, it also defines certain system variables that describe those data. For example, *_N* represents the total number of observations. *_n* represents the observation number: *_n* = 1 for the first observation, *_n* = 2 for the second, and so on to the last observation (*_n* = *_N*). If we issue a command such as the following, it creates a new variable, *caseID*, equal to the number of each observation as presently sorted:

```
. generate caseID = _n
```

Sorting the data another way will change each observation's value of *_n*, but its *caseID* value will remain unchanged. Thus, if we do sort the data another way, we can later return to the earlier order by typing

```
. sort caseID
```

Creating and saving unique case identification numbers that store the order of observations at an early stage of dataset development can facilitate later data management.

We can use explicit subscripts with variable names, to specify particular observation numbers. For example, the 4th observation in our global temperature dataset *global2.dta* is April 1880, with a temperature anomaly (*temp*) of $-.0912^{\circ}\text{C}$.

```
. display temp[4]
-.0912
```

Similarly, *temp[5]* is the temperature anomaly for May 1880, $-.151^{\circ}\text{C}$:

```
. display temp[5]
-.15099999
```

Explicit subscripting and the *_n* system variable have particular relevance when our data form a series. In this temperature example, either *temp* or, equivalently, *temp[_n]* denotes the value of the *_n*th observation. *temp[_n-1]* denotes the previous temperature, and *temp[_n+1]* denotes the next. Thus, we might define a new variable *diftemp*, which is equal to the change in *temp* since the previous month:

```
. generate diftemp = temp - temp[_n-1]
```

Chapter 12 on time series analysis returns to this topic.

Importing Data from Other Programs

Previous sections illustrated how to enter and edit data using the Data Editor. If our original data reside in an appropriately formatted spreadsheet, we just can copy and paste blocks of data from the spreadsheet into the empty Data Editor. Alternatively, Stata can import data from Excel spreadsheets directly through menu selections

File > Import > Excel spreadsheet (*.xls; *.xlsx)

or the **import excel** command. In the simplest case, we could import the first sheet in a spreadsheet file named *snowfall.xls* by typing the command

```
. import excel using C:\data\snowfall.xls, clear
```

But spreadsheets often contain titles, notes, subtables, multiple sheets, graphs or other features that complicate the process of reading them as data. To restrict the **import** operation to a particular range of cells, use a **cellrange()** option. The **sheet()** option can specify what sheet within the spreadsheet to import. A **firstrow** option tells Stata the first row of these cells contain variables names. For example, in spreadsheet *snowfall.xls* the first sheet, named “Berlin”, contains historical snowfall records for the town of Berlin, New Hampshire, as discussed in Hamilton et al. (2003). The data of interest reside in cells A5 through O56. Row 4 contains variable names.

```
. import excel using C:\data\snowfall.xls, sheet("Berlin")
  cellrange(a4:o56) firstrow clear
```

Although the **import excel** feature is fairly robust, some preparation of the Excel spreadsheet may speed the transition to an analyzable Stata dataset. For example, if there are variable names in the spreadsheet these should meet Stata criteria such as starting with a letter or underscore, and have no embedded blanks. Missing values should be replaced with blanks or numerical codes, and non-numeric characters removed from cells in columns meant to represent numerical variables.

Stata automatically decides whether each data column represents a numeric or string variable. If there are non-numeric values in a column, Stata takes that column to be a string variable, for which statistical calculations such as means and correlations will not be possible. If most of the values really are numeric, we could **generate** a new numerical version of that variable (with its actual string values coded as missing) using the **real()** function.

```
. generate newvar = real(oldvar)
```

Similar care is needed when we copy and paste from a spreadsheet into the Data Editor. Before selecting the block of data to copy, editing of the spreadsheet might be needed. One nice trick is to insert a row of variable names right above the top row of data in our spreadsheet. Then copy the row of names along with the rest of the data, and use Paste Special with Treat first row as variable names to place all of this information into an empty Data Editor.

The spreadsheet and Data Editor methods are quick and easy, but for larger projects it is important to have tools that work directly with computer files created by other statistical programs such as SAS or SPSS. SAS XPORT files can be imported through Stata menu choices

File > Import > SAS XPORT

or the **import sasxport** command. Other data formats can be read through the intermediate form of text files, or translated directly by a special third-party program.

We can illustrate text file methods with another climate-themed time series. El Niño–Southern Oscillation (ENSO) is a quasi-periodic climate pattern centered in the tropical Pacific Ocean but affecting other regions as well. The Multivariate ENSO Index (MEI) combines six observed variables describing tropical Pacific conditions (sea-level pressure, zonal and meridional surface winds, sea surface and surface air temperature, and cloudiness) into a single indicator for ENSO. Text file *MEI.raw* contains monthly values of MEI from January 1950 through December 2011. These are tab-separated values, a common format for text files written by spreadsheets. The first row of the text file contains a list of variable names: *mei1* for January MEI, *mei2* for February, and so forth (actually the “January” MEI value represents December–January, and February represents January–February, etc.). The first few rows of the text file look like this:

	year	mei1	mei2	mei3	mei4	mei5	mei6	mei7	mei8	mei9	mei10	mei11	mei12
1950	-1.022	-1.148	-1.287	-1.058	-1.423	-1.363	-1.342	-1.066	-5.576	-3.394	-1.154	-1.247	
1951	-1.068	-1.194	-1.216	-4.434	-2.264	.482	.756	.864	.779	.752	.728	.467	
1952	.406	.142	.096	.261	-.257	-.633	-.235	-.157	.362	.311	-.338	-.125	
1953	.024	.388	.272	.712	.833	.242	.421	.252	.522	.092	.049	.313	

We can read these data into Stata using the **insheet** command, with options to specify that values are tab-separated, and the first row contains variable names. After reading in the raw data we save them as a Stata-format file named *MEI0.dta*, which will be used again later.

```
. insheet using c:\data\MEI.raw, tab name clear
. save c:\data\MEI0.dta, replace
. describe

Contains data from c:\data\MEI0.dta
    obs:          63
    vars:         13
    size:      3,150
                                12 Feb 2012 09:40

variable   storage   display   value
name       type      format     label
year        int       %8.0g
mei1        float     %9.0g
mei2        float     %9.0g
mei3        float     %9.0g
mei4        float     %9.0g
mei5        float     %9.0g
mei6        float     %9.0g
mei7        float     %9.0g
mei8        float     %9.0g
mei9        float     %9.0g
mei10       float     %9.0g
mei11       float     %9.0g
mei12       float     %9.0g
```

Sorted by:

With a **comma** option instead of **tab**, **insheet** could read a text file of comma-separated values, which are another common spreadsheet output format. Text files can be read through Stata menus as well. Explore Data > Import to see the options available.

The examples so far assumed that raw data values are separated by commas, tabs or other known delimiters (which could be replaced with commas or tabs). A different arrangement called fixed-column format has values that are not necessarily delimited at all, but do occupy predefined column positions. The **infix** command can read such files. In the command syntax itself, or in a data dictionary existing in a separate file or as the first part of the data file, we have to specify exactly how the columns should be read.

Here is a simple example. Data exist in a text (ASCII) file named *nfresour.raw*:

```
198624087641691000
198725247430001044
198825138637481086
198925358964371140
1990    8615731195
1991    7930001262
```

These data concern natural resource production in Newfoundland. The four variables occupy fixed column positions: columns 1–4 are the years (1986...1991); columns 5–8 measure forestry production in thousands of cubic meters (2408...missing); columns 9–14 measure mine production in thousands of dollars (764,169...793,000); and columns 15–18 are the consumer

price index relative to 1986 (1000...1262). Notice that in fixed-column format, unlike whitespace or tab-delimited files, blanks indicate missing values, and the raw data contain no decimal points. To read *nfresour.raw* into Stata, we specify each variable's column position:

```
. infix year 1-4 wood 5-8 mines 9-14 CPI 15-18
      using nfresour.raw, clear
. list
```

	year	wood	mines	CPI
1.	1986	2408	764169	1000
2.	1987	2524	743000	1044
3.	1988	2513	863748	1086
4.	1989	2535	896437	1140
5.	1990	.	861573	1195
6.	1991	.	793000	1262

More complicated fixed-column formats might require a data dictionary. Data dictionaries can be straightforward, but they offer many possible choices. Type **help import** to see an outline of these commands. For more examples and explanation, consult the *User's Guide* and reference manuals. Stata also can load, write, or view data from ODBC (Open Database Connectivity) sources; see **help odbc**.

What if we need to export data from Stata to some other, non-ODBC program? **export excel** and **export sasxport** commands, or corresponding menu selections from

File > Export > Excel spreadsheet (*.xls; *.xlsx)
 File > Export > SAS XPORT

will write Excel spreadsheets or SAS XPORT files. The **outsheet** and **outfile** commands (or Data > Export) can write text files in several different formats. Another very quick possibility is to copy your data from Stata's Data Editor or Data Browser and paste this directly into a spreadsheet such as Excel. Often the best option, however, is to transfer data directly between the specialized system files saved by various spreadsheet, database or statistical programs. Some third-party programs perform such translations. StatTransfer, for example, will transfer data across many different formats including dBASE, Excel, FoxPro, Gauss, JMP, MATLAB, Minitab, OSIRIS, Paradox, R, S-Plus, SAS, SPSS, SYSTAT and Stata. Even large datasets hundreds of megabytes in size can be translated or excerpted quickly with this program. It is available through StataCorp (www.stata.com) or from its maker, Circle Systems (www.stattransfer.com). Transfer programs prove indispensable for analysts working in multi-program environments or exchanging data with colleagues.

One distinguishing feature of Stata is worth mentioning here. Stata datasets saved on one Stata platform (whether Windows, Mac or Unix) can be read without translation by Stata on any of the other platforms. To make a data file that can be read by an earlier version of Stata on any of these platforms, use the **saveold** command instead of **save**, or select

Save As > Save as type > Stata 9/10 Data

from the menus.

Combining Two or More Stata Files

We can combine Stata datasets in two general ways: **append** a second dataset that contains additional observations, or **merge** with other datasets that contain new variables or values. For example, file *lakewin1.dta* contains the ice-out dates for New Hampshire's largest lake, recorded by local observers over 121 years from 1887 through 2007.

```
. use C:\data\lakewin1.dta, clear
. describe

Contains data from C:\data\lakewin1.dta
obs:           121
vars:            3
size:        726

variable   storage    display      value
name      type       format     label
year      int         %ty        Year
winedate  int         %tdcymd   Date Lake Winnipesaukee Ice Out
winout    int         %9.0g     Lake Winnipesaukee Ice Out day

Sorted by: year
. list in -4/1



|      | year | winedate  | winout |
|------|------|-----------|--------|
| 118. | 2004 | 2004Apr20 | 111    |
| 119. | 2005 | 2005Apr20 | 110    |
| 120. | 2006 | 2006Apr3  | 93     |
| 121. | 2007 | 2007Apr23 | 113    |


```

In 2007, Lake Winnipesaukee ice out occurred on April 23, the 113th day of the year.

File *lakewin2.dta* contains newer data from 2008 through 2012. It has the same variables and format, so we can combine the update in *lakewin2.dta* with the older information in *lakewin1.dta*, using the **append** command.

```
. use C:\data\lakewin2.dta
(Lake Winnipesaukee ice out 2008-2012)

. describe
```

```
Contains data from C:\data\lakewin2.dta
obs: 5
vars: 3
size: 30
                                         Lake Winnipesaukee ice out
                                         2008-2012
                                         27 Mar 2012 09:48
```

variable	name	storage type	display format	value label	variable label
year		int	%ty		Year
winedate		int	%tdcymd		Date Lake Winnipesaukee Ice Out
winout		int	%9.0g		Lake Winnipesaukee Ice Out day

Sorted by: year

. list

	year	winedate	winout
1.	2008	2008Apr23	114
2.	2009	2009Apr12	102
3.	2010	2010Mar24	83
4.	2011	2011Apr19	109
5.	2012	2012Mar23	83

```
. append using c:\data\lakewin1
. sort year
. label data "Lake Winnipesaukee ice out 1887-2012"
. save c:\data\lakewin3
. list in -7/1
```

	year	winedate	winout
120.	2006	2006Apr3	93
121.	2007	2007Apr23	113
122.	2008	2008Apr23	114
123.	2009	2009Apr12	102
124.	2010	2010Mar24	83
125.	2011	2011Apr19	109
126.	2012	2012Mar23	83

In this example, both datasets contained the same variables, although that is not necessary for **append** to work. Variables that exist only in one of the appended datasets are assigned missing values for observations from the other dataset, when the two are combined.

append might be compared to lengthening a sheet of paper (that is, the dataset in memory) by taping a second sheet with new observations (rows) to its bottom. **merge**, in its simplest form, corresponds to widening our sheet of paper by taping a second sheet to its right side, thereby adding new variables (columns).

File *lakesun.dta* contains ice out dates for New Hampshire's second-largest lake over the years 1869 through 2012. Although the Lake Sunapee (*lakesun.dta*) and Lake Winnipesaukee (*lakewin3.dta*) records come from different sources, both form yearly series that could easily be combined into one dataset. We do this with the **merge 1:1 year** command.

. use C:\data\lakesun.dta

```
. describe
```

Contains data from C:\data\lakesun.dta
 obs: 144
 vars: 3
 size: 1,152

Lake Sunapee ice out 1869-2012
 27 Mar 2012 09:45

variable	storage type	display format	value label	variable label
year	int	%ty		Year
sunedate	float	%tdCYMD		Date Lake Sunapee Ice Out
sunout	int	%9.0g		Lake Sunapee Ice Out day

Sorted by: year

```
. merge 1:1 year using c:\data\lakewin3.dta
```

Result	# of obs.
not matched	18
from master	18
from using	0
matched	126
	(_merge==3)

Both datasets were already sorted by *year*; if they were not, we would have to **sort year** before merging. The **merge** results tell us that 126 years were present in both the “master” dataset (the data currently in memory—*lakesun.dta* in this example) and the “using” dataset (*lakewin3.dta*). A further 18 years (1869 to 1886) existed only in *lakesun.dta*, so the Lake Winnipesaukee variables will have missing values in those years. **merge** commands create a variable named *_merge* that records whether the observation came from the master data only (*_merge* = 1), the using data only (*_merge* = 2), or from both (*_merge* = 3). It is an important step to review *_merge* values carefully after each **merge** operation, making sure things turned out as planned. Before performing another **merge** operation, we must **drop** or **rename** *_merge*.

```
. drop _merge
. sort year
. list in 1/4
```

	year	sunedate	sunout	winedate	winout
1.	1869	1869May9	129	.	.
2.	1870	1870May9	129	.	.
3.	1871	1871April11	101	.	.
4.	1872	1872May2	123	.	.

```
. list in -4/1
```

	year	sunedate	sunout	winedate	winout
141.	2009	2009April11	101	2009April12	102
142.	2010	2010April3	93	2010Mar24	83
143.	2011	2011April21	111	2011April19	109
144.	2012	2012Mar22	82	2012Mar23	83

```
. label data "Sunapee & Winnipesaukee ice out 1869-2012"
. save C:\data\lakesunwin.dta, replace
```

In this example, we simply used **merge** to add new variables to our data, matching observations on *year*. By default, whenever the same variables are found in both datasets, those of the master data are retained and those of the using data ignored. The **merge** command has several options, however, that override this default. A command of the following form would allow any *missing values* in the master data to be replaced by corresponding nonmissing values found in the using data (here, *newdata.dta*):

```
. merge 1:1 year using newdata.dta, update
```

Or, a command such as the following causes *any values* from the master data to be replaced by nonmissing values from the using data, if the latter are different:

```
. merge 1:1 year using newdata, update replace
```

All of these examples show simple 1-to-1 merging. Also possible are 1-to-many (**1:m**), many-to-1 (**m:1**) or many-to-many (**m:m**) operations. Type **help merge** for details, and see the *Data Management* manual for further examples. Merging and appending data can be accomplished through Data > Combine datasets menus, as well.

Collapsing Data

Long after a dataset has been created, we might discover that for some purposes it has the wrong organization. Fortunately, several commands facilitate drastic restructuring of datasets. The simplest of these, **collapse**, aggregates data into means, medians or other statistics for groups defined by one or more variables. For illustration, we return to the data on monthly global temperatures from January 1880 to December 2011 (*global2.dta*), graphed earlier in Figure 2.1.

```
. use C:\data\global2.dta, clear
. describe

Contains data from C:\data\global2.dta
obs: 1,584                                         Global climate
vars: 4                                              12 Feb 2012 09:40
size: 14,256

variable name  storage type   display format   value label
year           int        %8.0g
month          byte       %8.0g
temp           float      %9.0g
edate          int        %tdmCY
                                         variable label
                                         Year
                                         Month
                                         NCDC global temp anomaly vs
                                         1901-2000, C
                                         elapsed date
                                         sorted by: edate
```

With **collapse**, we could build a simplified dataset containing mean temperature anomalies for 132 years instead of 1,584 separate months.

```
. collapse (mean) temp, by(year)
. label variable temp "NCDC annual mean temp anomaly, deg C"
. save C:\data\global_yearly.dta, replace
```

```
. describe
```

```
Contains data from C:\data\global_yearly.dta
obs:          132                               Global climate
vars:          2                                12 Feb 2012 10:02
size:         792

variable name  storage   display      value      variable label
              type       format     label
year           int        %8.0g
temp          float      %9.0g
                           Year
                           NCDC annual mean temp anomaly,
                           deg C

Sorted by: year
```

Our new annual dataset might be visualized with a spike plot, in which vertical spikes indicate distance of each year's temperature anomaly above or below the 1901–2000 mean.

```
. graph twoway spike temp year, xlabel(1880(20)2000)
```

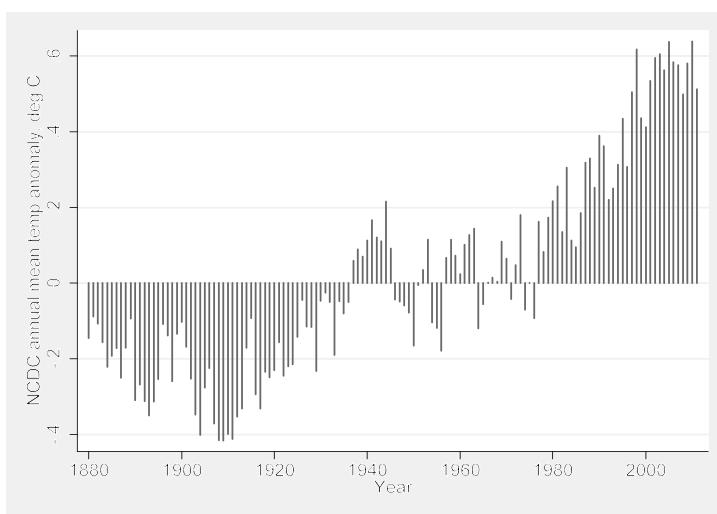


Figure 2.2

collapse can create variables based on any of the following summary statistics:

- mean** Means (default, if statistic is not specified)
- median** Medians
- p1** 1st percentiles
- p2** 2nd percentiles (and so forth to **p99**)
- sd** Standard deviations
- semean** Standard error of the mean ($sd/sqrt(n)$)
- sebinomial** Standard error of the mean, binomial ($sqrt(p(1-p)/n)$)
- sepoisson** Standard error of the mean, Poisson ($sqrt(mean)$)
- sum** Sums
- rawsum** Sums, ignoring optionally specified weight
- count** Number of nonmissing observations

max	Maximums
min	Minimums
iqr	Interquartile range
first	First values
last	Last values
firstnm	First nonmissing values
lastnm	Last nonmissing values

A wider range of statistics can be collected using the flexible **statsby** command, which works as a prefix for other analyses. In the following example we return to *global2.dta* and generate a new variable called *decade* (1880 for years 1880–1889, 1890 for 1890–1899, and so forth). Then we create a new dataset consisting of **summarize** statistics for temperature, by decade.

```
. use C:\data\global2.dta, clear
. gen decade = 10*int(year/10)
. statsby, by(decade) clear: summarize temp

(running summarize on estimation sample)

      command: summarize temp
          N: r(N)
      sum_w: r(sum_w)
      mean: r(mean)
      Var: r(Var)
      sd: r(sd)
      min: r(min)
      max: r(max)
      sum: r(sum)
      by: decade

Statsby groups
----- 1 ----- 2 ----- 3 ----- 4 ----- 5
.....
```

The new dataset contains number of observations, mean, variance, maximum and other **summarize** statistics for each decade. Figure 2.3 graphs the maximum monthly temperature anomaly (*max*) for each decade (setting aside the “2010” decade which just has two years).

```
. describe

Contains data
    obs:           14
    vars:            9
    size:          504
                                         statsby: summarize

      variable   storage   display      value
           name     type     format    label
-----  variable label
decade      float    %9.0g
N           float    %9.0g
sum_w       float    %9.0g
mean        float    %9.0g
Var          float    %9.0g
sd           float    %9.0g
min          float    %9.0g
max          float    %9.0g
sum          float    %9.0g

Sorted by:
Note: dataset has changed since last saved
```

```
. graph bar max if year<2010, over(decade)
    ytitle("Maximum global temperature anomaly, C")
```

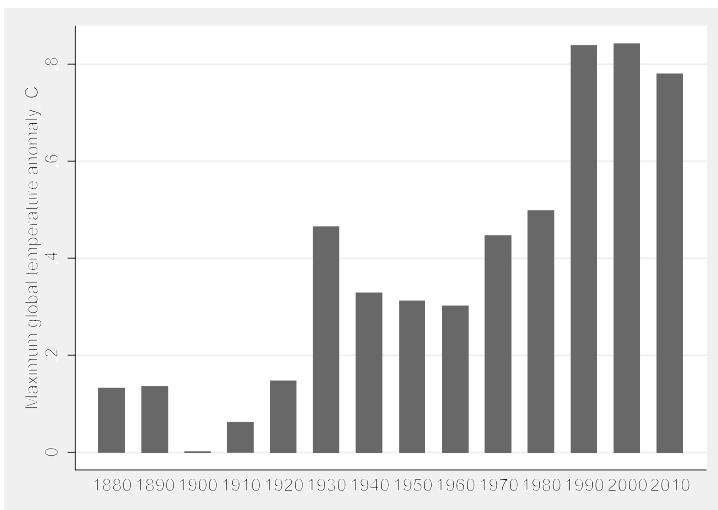


Figure 2.3

statsby can also make datasets of results from regression models or other analyses. Type **help statsby** or consult the *Data Management Reference Manual* for more information and examples.

Selecting

Statistics > Other > Collect statistics for a command across a by list

from the menus brings up the dialog box for this command. Another useful aggregation command, **contract**, creates a dataset that resembles a frequency table for any combinations of specified variables (see **help contract**).

Reshaping Data

A different sort of restructuring is possible through the **reshape** command. This command switches datasets between two basic configurations termed wide and long. Earlier in this chapter we built a dataset with the Multivariabe ENSO Index (*MEI0.dta*). The data are in wide format: years define the rows, but each month is a separate column. Thus, *mei1* represents the MEI value for January, *mei2* is February, and so on.

```
. use C:\data\MEI0.dta, clear
. describe
```

```

Contains data from C:\data\MEI0.dta
  obs:          62
  vars:         13
  size:      3,100
                                                               18 Feb 2012 10:46

```

variable name	storage type	display format	value label	variable label
year	int	%8.0g		
mei1	float	%9.0g		
mei2	float	%9.0g		
mei3	float	%9.0g		
mei4	float	%9.0g		
mei5	float	%9.0g		
mei6	float	%9.0g		
mei7	float	%9.0g		
mei8	float	%9.0g		
mei9	float	%9.0g		
mei10	float	%9.0g		
mei11	float	%9.0g		
mei12	float	%9.0g		

Sorted by:

```
. list year-mei7 in 1/5
```

	year	mei1	mei2	mei3	mei4	mei5	mei6	mei7
1.	1950	-1.022	-1.148	-1.287	-1.058	-1.423	-1.363	-1.342
2.	1951	-1.068	-1.194	-1.216	-.434	-.264	.482	.756
3.	1952	.406	.142	.096	.261	.257	-.633	-.235
4.	1953	.024	.388	.272	.712	.833	.242	.421
5.	1954	-.051	-.019	.169	-.504	-1.397	-1.578	-1.382

We can **reshape** these wide-format data into a time series in long format. The following command names a new variable to be created, *mei*. Each row of the new long-format dataset will have an observation identifier, i(*year*), and sub-observation identifier, j(*month*).

```
. reshape long mei, i(year) j(month)
(note: j = 1 2 3 4 5 6 7 8 9 10 11 12)
```

Data	wide	->	long
Number of obs.	62	->	744
Number of variables	13	->	3
j variable (12 values)		->	month
xij variables:	mei1 mei2 ... mei12	->	mei

```
. compress
. sort year month
. label variable mei "Multivariate ENSO Index"
. save C:\data\mei1.dta, replace
. describe
```

```

Contains data from C:\data\mei1.dta
obs: 744
vars: 3
size: 5,208
18 Feb 2012 10:51
variable name  storage  display      value
              type    format   label
year           int     %8.0g
month          byte    %9.0g
mei            float   %9.0g
                                         Multivariate ENSO Index
Sorted by: year month
.list in 1/5



|    | year | month | mei    |
|----|------|-------|--------|
| 1. | 1950 | 1     | -1.022 |
| 2. | 1950 | 2     | -1.148 |
| 3. | 1950 | 3     | -1.287 |
| 4. | 1950 | 4     | -1.058 |
| 5. | 1950 | 5     | -1.423 |


```

Now we have the Multivariate ENSO Index time series in year/month form, similar to the year/month time series of global surface temperatures (*global2.dta*) we built earlier. With both datasets sorted by year and month, we can **merge** the two into a common file.

```

.use C:\data\global2.dta, clear
.merge 1:1 year month using c:\data\mei1.dta

Result                                # of obs.
not matched
    from master                         840
    from using                           840 (_merge==1)
                                         0   (_merge==2)
matched                               744   (_merge==3)

```

The temperature data in *global2.dta* cover each month from January 1880 through December 2011, whereas *mei1.dta* covers only January 1950 through December 2011. Consequently, $70 \times 12 = 840$ months exist only in the master data and are not matched; the remaining $12 \times 62 = 744$ months exist in both datasets and are matched one to one.

After saving the new merged data as *global3.dta*, we can draw a time plot with both temperature and *mei* over the years 1950–2011. These two variables have different scales, so *mei* is assigned to the right-hand *y* axis, denoted **yaxis(2)**. The **graph** command below overlays two line plots, one for *temp* and one for *mei*, the latter drawn with a dashed line. The command also specifies a legend with two rows, instead of the default here which would be two columns. A first look at the graph suggests that global temperature and the ENSO index often vary together from year to year, but ENSO lacks the decadal upward trend of temperature. Chapter 12 applies time series modeling for a more rigorous analysis of this point.

```

.sort year month
.drop _merge
.compress

```

```
. save c:\data\global3.dta, replace
. graph twoway line temp edate ||
    line mei edate, yaxis(2) lpattern(dash) ||
    if year > 1949, legend(row(2))
```

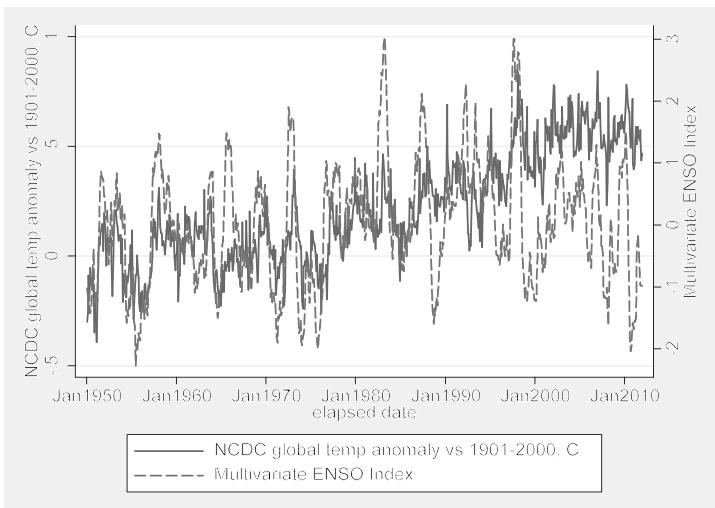


Figure 2.4

reshape works equally well in reverse, to switch data from long to wide format. We could convert the year/month time series of temperature and MEI into a wide dataset in which each row was a year, and each column a variable/month, by the following commands (not shown).

```
. drop edate
. reshape wide mei temp, i(year) j(month)
```

Using Weights

Stata understands four types of weighting:

- aweight** Analytical weights, used in weighted least squares (WLS) regression and similar procedures.
- fweight** Frequency weights, counting the number of duplicated observations. Frequency weights must be integers.
- iweight** Importance weights, however you define importance.
- pweight** Probability or sampling weights, proportional to the inverse of the probability that an observation is included due to sampling strategy.

Not all types of weighting have been defined for all types of analyses. We cannot, for example, use **pweight** with the **tabulate** command. Using weights effectively requires a clear understanding of what we want them to accomplish in a particular analysis.

Weights have many statistical applications, including methods of compensating for originally disproportionate or complex sampling designs — a common feature of surveys. **pweight** provides one way to adjust for sampling bias, using probability weights proportional to 1/(probability of selection). Analysis of survey data using probability weights is a particular strength of Stata, introduced in Chapter 4.

In some instances, weighting involves something simpler — an aggregate dataset in which the variables are statistics summarizing many individual observations. For example, dataset *Nations2.dta* contains United Nations human-development indicators that characterize living conditions in 194 nations.

Contains data from C:\data\Nations2.dta				UN Human Development Indicators
obs:	194 <th>vars:</th> <td>13</td> <th>1 Apr 2012 11:30</th>	vars:	13	1 Apr 2012 11:30
size:	12,804 <th></th> <th></th> <th></th>			
variable	storage type	display format	value label	variable label
country	str21	%21s		Country
region	byte	%8.0g		Region
gdp	float	%9.0g	region	Gross domestic product per cap 2005\$, 2006/2009
school	float	%9.0g		Mean years schooling (adults) 2005/2010
adfert	float	%8.0g		Adolescent fertility: births/1000 fem 15-19, 2010
chdmort	float	%9.0g		Prob dying before age 5/1000 live births 2005/2009
life	float	%9.0g		Life expectancy at birth 2005/2010
pop	float	%9.0g		Population 2005/2010
urban	float	%9.0g		Percent population urban 2005/2010
femlab	float	%9.0g		Female/male ratio in labor force 2005/2009
literacy	float	%9.0g		Adult literacy rate 2005/2009
co2	float	%9.0g		Tons of CO2 emitted per cap 2005/2006
gini	float	%9.0g		Gini coef income inequality 2005/2009

Sorted by: region country

The mean life expectancy is 68.7 years:

variable	obs	mean	std. dev.	min	max
life	194	68.7293	10.0554	45.85	82.76666

The mean above represents the average life expectancy for the 194 nations in the sample, rather than the average life expectancy for the 7 billion people who live in those nations. That is, it weights the life expectancy of the smallest nation (Tuvalu, a Pacific island nation with about 10,000 people) the same as the life expectancy of the largest (China, population about 1.3 billion). Using population as a frequency weight, we get a better estimate of the mean life expectancy for all 7 billion people.

```
. summarize life [fweight = pop]
```

Variable	Obs	Mean	Std. Dev.	Min	Max
life	6.669e+09	68.93644	8.095538	45.85	82.76666

Probability weights (**pweight**) get more attention in Chapter 4. Analytical weights (**aweight**) are useful in graphing (Chapter 3) and for weighted least squares (Chapters 7, 8), among other things. Importance weights (**iweight**) have no fixed definition, but could be applied in programs written for special purposes.

Creating Random Data and Random Samples

The pseudo-random number function **runiform()** lies at the heart of Stata's ability to generate random data or to sample randomly from the data at hand. The *Base Reference Manual* (Functions) provides a technical description of this 32-bit pseudo-random generator. If we presently have data in memory, then a command such as the following creates a new variable named *randnum*, having apparently random 16-digit values over the interval [0,1).

```
. generate randnum = runiform()
```

We could also create a random dataset from scratch. To do so we first clear other data from memory (if they were valuable, **save** them first). Next, set the number of observations desired for the new dataset. Explicitly setting the seed number makes it possible to later reproduce the same “random” results. Finally, we generate our random variable. The following commands create a dataset with 10 observations and one variable, called *randnum*.

```
. clear
. set obs 10
. set seed 12345
. generate randnum = runiform()
. list
```

	randnum
1.	.309106
2.	.6852276
3.	.1277815
4.	.5617244
5.	.3134516
6.	.5047374
7.	.7232868
8.	.4176817
9.	.6768828
10.	.3657581

In combination with Stata's algebraic, statistical and special functions, **runiform()** can simulate values sampled from a variety of theoretical distributions. If we want *newvar* sampled from a uniform distribution over [0,428) instead of the usual [0,1), we type

```
. generate newvar = 428 * runiform()
```

These will still be 16-digit values. Perhaps we want only integers from 1 to 428 (inclusive). The ceiling or `ceil()` function provides a simple way to do this:

```
. generate newvar = ceil(428 * runiform())
```

To simulate 1,000 throws of a six-sided die, type

```
. clear
. set obs 1000
. generate roll = ceil(6 * runiform())
. tabulate roll
```

roll	Freq.	Percent	Cum.
1	170	17.00	17.00
2	167	16.70	33.70
3	149	14.90	48.60
4	171	17.10	65.70
5	166	16.60	82.30
6	177	17.70	100.00
Total	1,000	100.00	

We theoretically expect 16.67% ones, 16.67% twos and so on, but in any one sample like these 1,000 “throws,” the observed percentages will vary randomly around their expected values.

To simulate 1,000 throws of a pair of six-sided dice, type

```
. generate dice = ceil(6 * runiform()) + ceil(6 * runiform())
. tabulate dice
```

dice	Freq.	Percent	Cum.
2	27	2.70	2.70
3	62	6.20	8.90
4	78	7.80	16.70
5	120	12.00	28.70
6	154	15.40	44.10
7	147	14.70	58.80
8	145	14.50	73.30
9	97	9.70	83.00
10	89	8.90	91.90
11	52	5.20	97.10
12	29	2.90	100.00
Total	1,000	100.00	

We can use `_n` to begin an artificial dataset as well. The following commands create a new 5,000-observation dataset with one variable named `index`, containing values from 1 to 5,000.

```
. set obs 5000
. generate index = _n
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
roll	1000	3.527	1.732129	1	6
dice	1000	6.948	2.428414	2	12
index	5000	2500.5	1443.52	1	5000

To generate random variables from a normal (Gaussian) distribution, use the function **rnormal()**. The following example creates a dataset with 2,000 observations and 2 variables: z from an $N(0,1)$ population, and u from $N(500,75)$.

```
. clear
. set obs 2000
. generate z = rnormal()
. generate u = rnormal(500,75)
```

The actual sample means and standard deviations differ slightly from their theoretical values:

```
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
z	2000	-.0022046	1.013775	-3.645242	3.620961
u	2000	498.4104	73.56337	248.0803	739.0969

If z follows a normal distribution, $v = e^z$ follows a lognormal distribution. To form a lognormal variable v based upon a standard normal distribution,

```
. generate v = exp(rnormal())
```

Taking logarithms, of course, normalizes a lognormal variable.

To simulate w values drawn randomly from an exponential distribution with mean and standard deviation $\mu = \sigma = 3$,

```
. generate w = -3 * ln(runiform())
```

For other means and standard deviations, substitute other values for 3.

$x5$ follows a χ^2 (chi-squared) distribution with five degrees of freedom:

```
. generate x5 = rchi2(5)
```

y follows a binomial distribution, given 10 trials and success probability .2:

```
. generate y = rbinomial(10,.2)
```

$t45$ follows a Student's t distribution with 45 degrees of freedom:

```
. generate t45 = rt(45)
```

Type **help random** to see a list of other available functions for generating random variables from beta, gamma, hypergeometric, negative binomial or Poisson distributions.

The **drawnorm** command provides an alternative way to generate multiple normal variables, and optionally to specify the correlations between them. Using **drawnorm** to generate 5,000 observations of just one variable from N(0,1), type

```
. clear
. drawnorm z, n(5000)
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
z	5000	-.0041833	1.008007	-3.584255	3.630447

Below, we will create three further variables. Variable *x1* is from an N(0,1) population, variable *x2* is from N(100,15), and *x3* is from N(500,75). Furthermore, we define these variables to have the following population correlations:

	x1	x2	x3
x1	1	0.4	-0.8
x2	0.4	1	0
x3	-0.8	0	1

The procedure for creating such data requires first defining the correlation matrix *C*, and then using *C* in the **drawnorm** command:

```
. mat C = (1, .4, -.8 \ .4, 1, 0 \ -.8, 0, 1)
. drawnorm x1 x2 x3, means(0,100,500) sds(1,15,75) corr(C)
. summarize x1-x3
```

Variable	Obs	Mean	Std. Dev.	Min	Max
x1	5000	-.0289361	1.001212	-3.628264	3.889191
x2	5000	99.82377	14.86525	39.83408	153.0636
x3	5000	502.0403	75.24015	223.4024	752.7481

```
. correlate x1-x3
(obs=5000)
```

	x1	x2	x3
x1	1.0000		
x2	0.4004	1.0000	
x3	-0.8031	-0.0005	1.0000

Compare the sample variables' correlations and means with the theoretical values given earlier. Random data generated in this fashion can be viewed as samples drawn from theoretical populations. We should not expect the samples to have exactly the theoretical population parameters (in this example, an *x3* mean of 500, *x1*–*x2* correlation of 0.4, *x1*–*x3* correlation of -0.8, and so forth). Artificial uncorrelated or correlated datasets also can be created via menus and dialog boxes, under

Statistics > Other > Draw a sample from a normal distribution
or
Statistics > Other > Create a dataset with specified correlation structure

The command **sample** makes unobtrusive use of **runiform**'s random generator to obtain random samples of the data in memory. For example, to discard all but a 10% random sample of the original data, type

```
. sample 10
```

When we add an **in** or **if** qualifier, **sample** applies only to those observations meeting our criteria. For example,

```
. sample 10 if age < 26
```

would leave us with a 10% sample of those observations with *age* less than 26, plus 100% of the original observations with *age* \geq 26.

We could also select random samples of a particular size. To discard all but 90 randomly-selected observations from the dataset in memory, type

```
. sample 90, count
```

The section in Chapter 14 on Monte Carlo simulations provides further examples of random variable generation.

Writing Programs for Data Management

Data management on larger projects often involves repetitive or error-prone tasks that are best handled by writing specialized Stata programs. Advanced programming can become very technical, but we could also begin by writing simple programs that consist of nothing more than a sequence of Stata commands, typed and saved as a text file. Text files can be created using your favorite word processor or text editor, which should offer several kinds of text files among its options under **File > Save As**. One convenient way to create such text files is through Stata's Do-file Editor, which is brought up by clicking **Window > Do-file Editor** or the icon . Alternatively, bring up the Do-file Editor by typing the command **doedit**, or **doedit filename** if *filename* exists. Commands in the Review window can be highlighted and sent directly to the Do-File Editor (right-click to get this menu choice). Commands can also be copied and pasted into the Do-File Editor from other sources such as log files or the Results window.

Across several sections of this chapter we began building a global climate dataset, starting with temperature, then reshaping and merging the Multivariate El Niño/Southern Oscillation Index (MEI), and finally graphing temperature and MEI together as Figure 2.4. The commands that executed each of these steps could be assembled into a single do-file, as follows. Note the use of **///** to continue the long **graph twoway** command onto more than one line. At its end, this do-file saves Figure 2.4 in Stata graph (.gph) and enhanced Windows metafile (.emf) file formats.

```
insheet using C:\data\global.csv, comma clear
label data "Global climate"
label variable year "Year"
label variable month "Month"
```

```

label variable temp "NCDC global temp anomaly vs 1901-2000, C"
generate edate = mdy(month, 15, year)
label variable edate "elapsed date"
format edate %tdmCY
sort year month
order year month edate
save C:\data\global2.dta, replace
use C:\data\MEI0.dta, clear
reshape long mei, i(year) j(month)
sort year month
label variable mei "Multivariate ENSO Index"
save C:\data\mei1.dta, replace
use C:\data\global2.dta, clear
merge 1:1 year month using c:\data\mei1.dta
sort year month
drop _merge
compress
save c:\data\global3.dta, replace
graph twoway line temp edate ///
    || line mei edate, yaxis(2) lpattern(dash) ///
    || if year>1949, legend(row(2))
graph save Graph "C:\graphs\fig02_04.gph", replace
graph export "C:\graphs\fig02_04.emf", as(emf) replace

```

This file could be written by highlighting commands in the Review window, then right-click and Send to Do-file Editor. Save the do-file with a new name, such as *global.do*. Once the do-file is created, we can run it by selecting File > Do and opening *global.do* from the menus; or just by typing a command such as

```
. do global
```

Such batch-mode programs are usually saved with a .do extension. More elaborate programs (defined by either do-files or automatic ado-files) can be stored in memory, and can call other programs in turn, creating new Stata commands and opening a world of possibilities for the adventurous.

Stata ordinarily interprets the end of a command line as the end of that command. This is reasonable onscreen, where the line can be arbitrarily long, but does not work as well when we are typing commands in a text file. Three forward slashes (///) at the end of a physical line tell Stata that the command is continued on the next physical line. The command executes only after reaching a line that does not end with ///

Another way to handle long lines in do-files is to use a #delimit ; command, which sets a semicolon as the end-of-command delimiter. In the example below we make a semicolon the delimiter, type a long command that does not end until a semicolon appears, and then finally reset the delimiter to its usual value, a carriage return (cr)

```

#delimit ;
graph twoway line temp edate
    || line mei edate, yaxis(2) lpattern(dash)
    || if year>1949, legend(row(2)) ;
#delimit cr

```

Stata normally pauses each time the Results window becomes full of information, and waits to proceed until we press the space bar or any other key (or click ). Instead of pausing, we can ask Stata to continue scrolling until the output is complete. Typed in the Command window or as part of a program, the command

```
. set more off
```

calls for continuous scrolling. This is convenient if our program produces much screen output that we don't want to see, or if it is writing to a log file that we will examine later. Typing

```
. set more on
```

returns to the usual mode of waiting for keyboard input before scrolling.

Graphs

Graphs appear in every chapter of this book — one indication of their value and integration with other analyses in Stata. Analytical graphics have always been one of Stata’s strengths, and reason enough for many users to choose Stata over other packages. Attractive and publishable basic graphs are easy to draw, using commands or choices from the menus under Graphics. Users who imagine more elaborate or creative graphs will find their efforts supported by an impressive array of tools and options, described in the *Graphics Reference Manual*, and illustrated by many examples in *A Visual Guide to Stata Graphics* (Mitchell 2012).

In this chapter we introduce a selection of Stata’s main graph types, taking an example- rather than syntax-oriented approach (see the *Graphics Reference Manual* or **help** files for much more about syntax). Some of our examples are quite basic, using few or no options, but often the basic examples are followed by others illustrating options and tricks to make publication-ready graphs. Although the chapter can be read in a linear fashion, working out each example in turn, it is also meant to serve as a gallery through which one could browse for images and ideas.

The full range of graphical choices goes far beyond what this book can cover, but these examples point out a few of the possibilities. Later chapters supply further illustrations. The Graphics menu provides point-and-click access to most graphing procedures. Experimenting with dialog boxes and using the Submit button is a good way to learn what is available, and particularly the many choices for **twoway** graphs.

Example Commands

- **histogram y, frequency**

Draws histogram of variable *y* , showing frequencies on the vertical axis.

- **histogram y, start(0) width(10) norm fraction**

Draws histogram of *y* with bins 10 units wide, starting at 0. Adds a normal curve based on the sample mean and standard deviation, and shows fraction of the data on the vertical axis.

- **histogram y, by(x, total) percent**

In one figure, draws separate histograms of *y* for each value of *x*, and also a “total” histogram for the sample as a whole. Shows percentages on the vertical axis.

- **kdensity x, generate(xpoints xdensity) width(20) biweight**

Produces and graphs kernel density estimate of the distribution of *x*. Two new variables are created: *xpoints* containing the *x* values at which the density is estimated, and *xdensity* with the density estimates themselves. **width(20)** specifies the halfwidth of the kernel, in units of the variable *x*. If **width()** is not specified, the default follows a simple formula for optimal. The **biweight** option in this example calls for a biweight kernel, instead of the default **epanechnikov**.

- **graph twoway scatter y x**

Displays a basic two-variable scatterplot of *y* against *x*. The **graph** part is optional for all **twoway** family commands; for example, we could type **twoway scatter y x**.

- **graph twoway lfit y x || scatter y x**

Visualizes the linear regression of *y* on *x* by overlaying two **twoway** graphs: the regression (linear fit or **lfit**) line, and the *y* vs. *x* scatterplot To include a 95% confidence band for the regression line, replace **lfit** with **lfitci**.

- **graph twoway scatter y x, xlabel(0(10)100) ylabel(-3(1)6, horizontal)**

Constructs scatterplot of *y* vs. *x*, with *x* axis labeled at 0, 10, ..., 100. *y* axis is labeled at -3, -2, ..., 6, with labels written horizontally instead of vertically (the default).

- **graph twoway scatter y x, mlabel(country)**

Constructs scatterplot of *y* vs. *x*, with data points (markers) labeled by the values of variable *country*.

- **graph twoway scatter y x1, by(x2)**

In one figure, draws separate *y* vs. *x1* scatterplots for each value of *x2*.

- **graph twoway scatter y x1 [fweight = population], msymbol(Oh)**

Draws a scatterplot of *y* vs. *x1*. Marker symbols are hollow circles (**Oh**), with their area proportional to frequency-weight variable *population*.

- **graph twoway connect y time**

A basic time plot of *y* against *time*. Data points are shown connected by line segments. To show line segments but no data-point markers, use **line** instead of **connect**:

graph twoway line y time

An alternative, simpler time plot command **tsline** works with **tsset** data, where the time variable has been declared (see Chapter 12):

tsline y

- **graph twoway line y1 y2 time**

Draws a time plot (in this example, a line plot) with two *y* variables that both have the same scale, and are graphed against an *x* variable named *time*.

- **graph twoway line y1 time, yaxis(1) || line y2 time, yaxis(2)**

Draws a time plot with two *y* variables that have different scales, by overlaying two individual line plots. The left-hand *y* axis, **yaxis(1)**, gives the scale for *y1*, while the right-hand *y* axis, **yaxis(2)**, gives the scale for *y2*.

- . **graph twoway contour temperature y x**
Draws a contour plot displaying *temperature* as filled contours in (*y*, *x*), such as (*latitude*, *longitude*). Options control details such as number of contours, interpolation methods and colors.
- . **graph matrix x1 x2 x3 x4 y**
Constructs a scatterplot matrix, showing all possible scatterplot pairs among the variables listed.
- . **graph box y1 y2 y3**
Constructs box plots of variables *y1*, *y2*, and *y3*.
- . **graph box y, over(x) yline(23)**
Constructs box plots of *y* for each value of *x*, and draws a horizontal line at *y* = 23.
- . **graph pie a b c**
Draws one pie chart with slices indicating the relative amounts of variables *a*, *b*, and *c*. The variables must have similar units.
- . **graph bar (sum) a b c**
Shows the sums of variables *a*, *b*, and *c* as side-by-side bars in a bar chart. To obtain means instead of sums, type **graph bar (mean) a b c**. Further options include bars representing medians, percentiles, counts, or other statistics for each variable (same options as **collapse**).
- . **graph bar (mean) a, over(x)**
Draws a bar chart showing the mean of variable *a* at each value of variable *x*.
- . **graph bar (asis) a b c, over(x) stack**
Draws a bar chart in which the values (“as is”) of variables *a*, *b* and *c* are stacked on top of one another, at each value of variable *x*.
- . **graph dot (median) y, over(x)**
Draws a dot chart, in which dots along a horizontal scale mark the median value of *y* at each level of *x*. **graph dot** supports the same statistical options as **graph bar** or **collapse**.
- . **qnorm y**
Draws a quantile–normal plot (normal probability plot) showing quantiles of *y* versus corresponding quantiles of a normal distribution.
- . **rchart x1 x2 x3 x4 x5, connect(1)**
Constructs a quality-control R chart graphing the range of values represented by variables *x1*–*x5*. Type **help qc** to see the full range of quality-control graphs Stata offers. These can also be accessed through menus: **Graphics > Quality control**.

Graph options, such as those controlling titles, labels, and tick marks on the axes, are common across graph types wherever this makes sense. Moreover, the underlying logic of Stata’s graph commands is consistent from one type to the next. These common elements are the key to gaining graph-building fluency, as the basics fall into place.

Histograms

Histograms, displaying the distribution of measurement variables, are most easily produced with their own command **histogram**. For examples, we return to the data on 194 nations seen earlier in Chapter 2, containing human-development indicators gathered by the United Nations.

Contains data from C:\data\Nations2.dta				UN Human Development Indicators
obs:	194 <th>vars:</th> <td>13</td> <th>1 Apr 2012 11:30</th>	vars:	13	1 Apr 2012 11:30
size:	12,804 <th></th> <th></th> <th></th>			
variable	storage type	display format	value label	variable label
country	str21	%21s		Country
region	byte	%8.0g		Region
gdp	float	%9.0g	region	Gross domestic product per cap 2005\$, 2006/2009
school	float	%9.0g		Mean years schooling (adults) 2005/2010
adfert	float	%8.0g		Adolescent fertility: births/1000 fem 15-19, 2010
chdmort	float	%9.0g		Prob dying before age 5/1000 live births 2005/2009
life	float	%9.0g		Life expectancy at birth 2005/2010
pop	float	%9.0g		Population 2005/2010
urban	float	%9.0g		Percent population urban 2005/2010
femlab	float	%9.0g		Female/male ratio in labor force 2005/2009
literacy	float	%9.0g		Adult literacy rate 2005/2009
co2	float	%9.0g		Tons of CO2 emitted per cap 2005/2006
gini	float	%9.0g		Gini coef income inequality 2005/2009

Sorted by: region country

Figure 3.1 shows a simple histogram of *adfert*, the adolescent fertility rate. It was produced by the following command:

```
. histogram adfert, percent
```

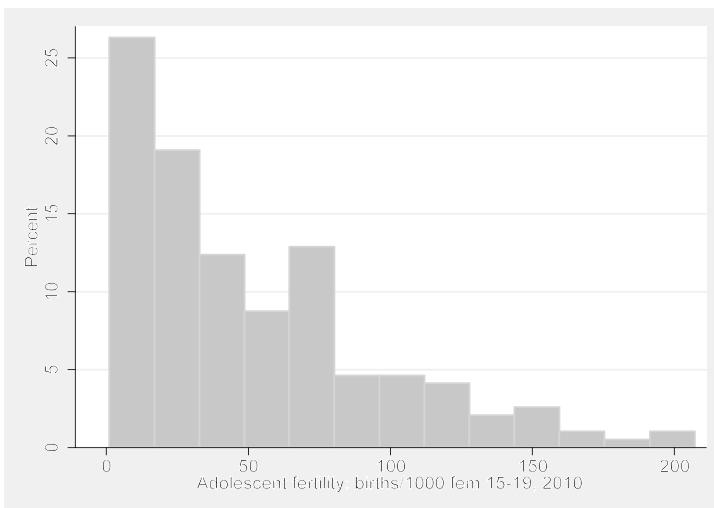


Figure 3.1

Under the Prefs > Graph Preferences menus, we have the choice of several pre-designed schemes for the default colors and shading of our graphs. Custom schemes can be defined as well. Most examples in this book employ the s2color scheme, which among other things calls for shaded margins around each graph. Experimenting with the different monochrome and color schemes helps to determine which works best for a particular purpose. A graph drawn and saved under one scheme can subsequently be retrieved and re-saved under a different one, as described later.

Options can be listed in any order following the comma in a graph command. Figure 3.1 illustrates one option: **percent** (instead of **density**, the default) is shown on the vertical axis. Once a graph is onscreen, menu choices provide the easiest way to print it, save it to disk, or copy and paste it into another program such as a word processor.

Figure 3.1 reveals the positive skew of this distribution, with a mode not far above 0 and an upper limit around 200. It is hard to describe the graph more specifically because the bars do not line up with x-axis tick marks. Figure 3.2 contains a version of the same histogram but with some optional improvements:

- frequency** Frequencies are shown on the vertical (y) axis.
- start(0)** The histogram's first bar (bin) starts at 0.
- width(10)** The width of each bar (bin) is 10.
- xlabel(0(20)200)** The x axis is labeled from 0 to 200, in increments of 20.
- xtick(10(20)210)** The x axis has tick marks from 10 to 210, in increments of 20.
- ylabel(0(2)12, grid gmax)** The y axis is labeled from 0 to 35, in increments of 5. A grid of horizontal lines is drawn, including a line at the maximum value.
- title("Adolescent fertility rate in 194 nations")** The graph has a title at top.

The command below is shown as four lines to make it easier to read. To make this four-line command work in a do-file, we could add /// to the ends of the first three lines, indicating the command continues on the next physical line.

```
. histogram adfert, frequency start(0) width(10)
    xlabel(0(20)200) xtick(10(20)210)
    ylabel(0(5)35, grid gmax)
    title("Adolescent fertility rate in 194 nations")
```

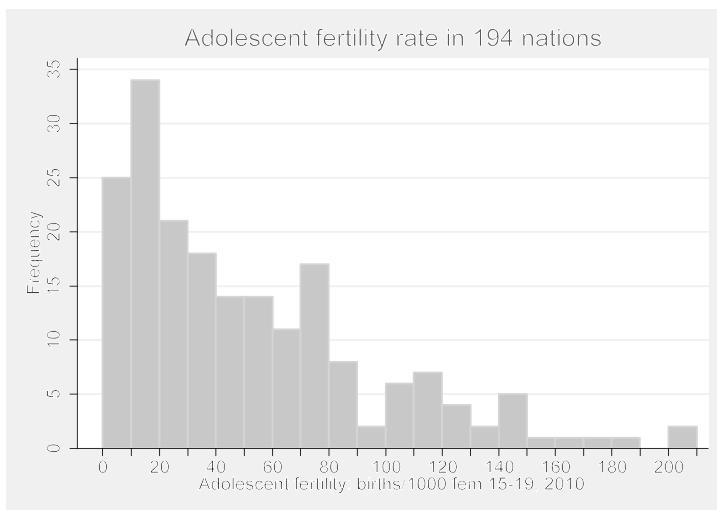
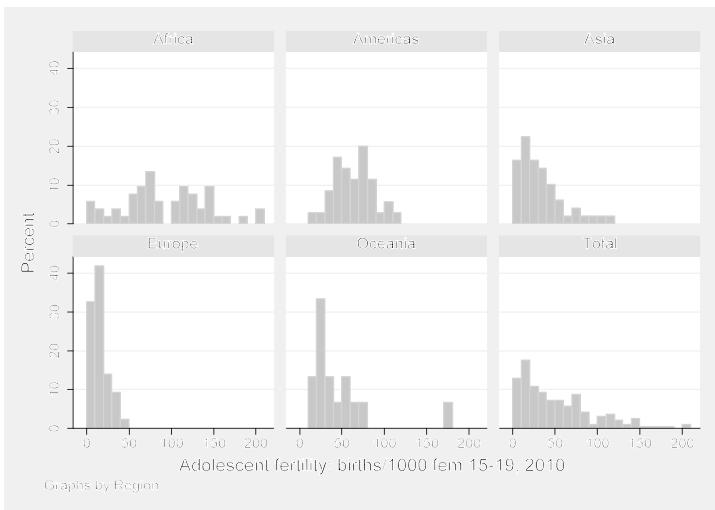


Figure 3.2

Figure 3.2 helps us to describe the distribution more specifically. For example, we now see that in 34 nations, the adolescent fertility rates are between 10 and 20. Type **help histogram** to see a complete list of options and syntax for this command. There also exists a separate command, **twoway histogram**, that draws histograms allowing other options common to the **twoway** family of graphs discussed later. You can learn about it by typing **help twoway histogram**.

One option that **histogram** shares with other graph types is the ability to create multiple small graphs for each value of a specified variable, using a **by(varname)** option. Figure 3.3 illustrates with histograms of *adfert* for each of the five regions, along with a sixth (**total**) histogram showing the distribution for all regions.

```
. histogram adfert, percent start(0) width(10) by(region, total)
```

**Figure 3.3**

Box Plots

Box plots convey information about center, spread, symmetry and outliers at a glance. For example, Figure 3.4 is a simple box plot of *adfert* (adolescent fertility rate) obtained by typing

```
. graph box adfert
```

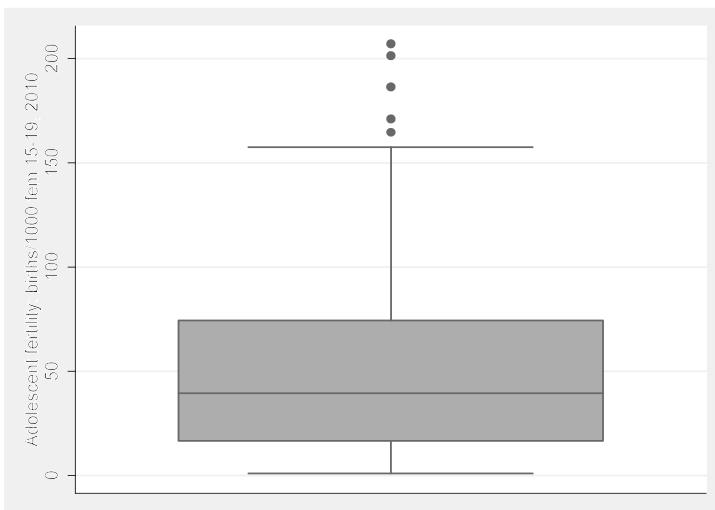
**Figure 3.4**

Figure 3.4 confirms the positive skew of this distribution, and shows five high outliers. The box in a box plot extends from approximate first to third quartiles, a distance called the interquartile range (IQR). It therefore contains roughly the middle 50% of the data. (Stata's box plots define quartiles in the same manner as **summarize, detail**.) Outliers, defined as observations more than 1.5(IQR) beyond the first or third quartile, are plotted as individual points.

Figure 3.5 identifies the *ad fert* outliers by labeling their markers with values of variable *country* (country names). It also specifies a non-default title for the *y* axis. The **marker** option can control the symbols and other properties denoting outliers as well. Specifying **marker(1)** in this example means that this option refers to the first-named *y* variable. There is only one *y* variable here, but in other cases we could have two or more, and mark their outliers in distinct ways.

```
. graph box ad fert, marker(1, mlabel(country))
    ytitle("Births per 1,000 females 15-19")
```

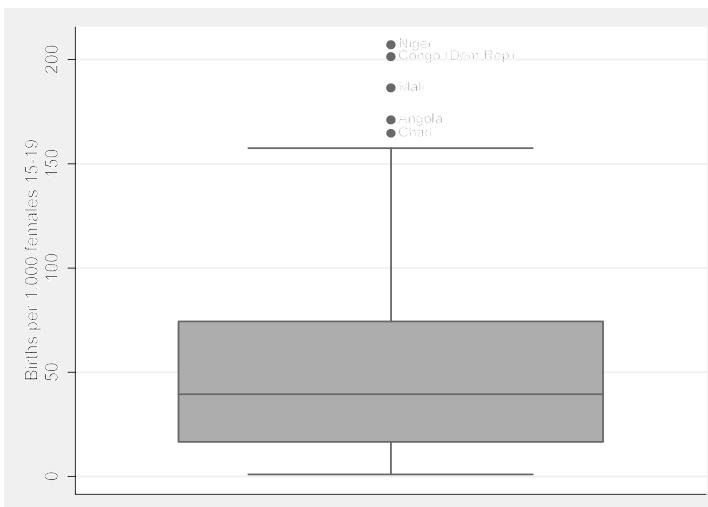


Figure 3.5

One of the most common applications for box plots involves comparing the distribution of one variable over categories of a second. Figure 3.6 compares the distribution of *ad fert* across *region*. The overall median is indicated by a horizontal line placed by the **yline(39.3)** option.

```
. graph box adfert, marker(1, mlabel(country)) yline(39.3) over(region)
```

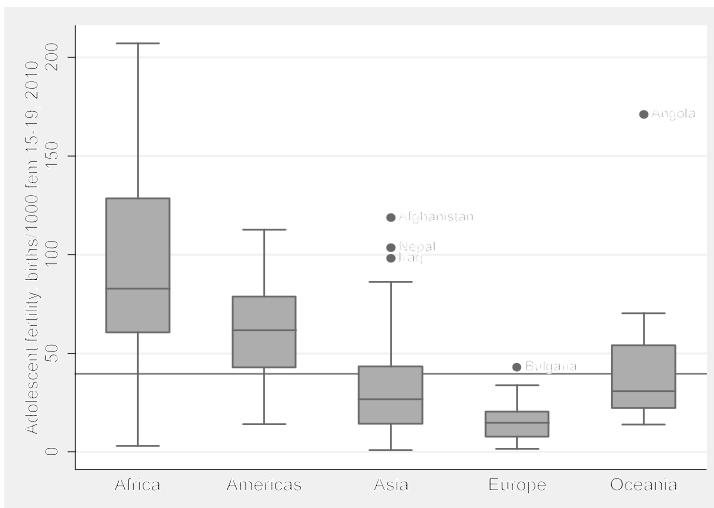


Figure 3.6

Box plots can have a horizontal orientation instead of vertical, via the **graph hbox** command. Figure 3.7 illustrates using per capita carbon dioxide emissions (*co2*), another variable from the *Nations2.dta* dataset. This example also shows off several title or labeling options, which could be applied to any type of graph. The **note()** and **caption()** options place text below the graph. In this figure, “Statistics with Stata” appears in bold, and “Example of horizontal box plots” in italicics. In the **ytitle** (y axis title, which in a horizontal box plot refers to the horizontal axis), CO₂ is given its proper subscript. Bold, italic, subscript and other text attributes are controlled within graphs using Stata markup and control language (SMCL) features. Type **help graph text** to see other possibilities and examples.

```
. graph hbox co2, over(region)
    note("note: {bf:Statistics with Stata}, version 12")
    caption("caption: United Nations Human Development Report 2011")
    title("title: {it:Example of horizontal box plots}")
    ytitle("ytitle: Tons of CO{subscript:2} emitted per capita")
```

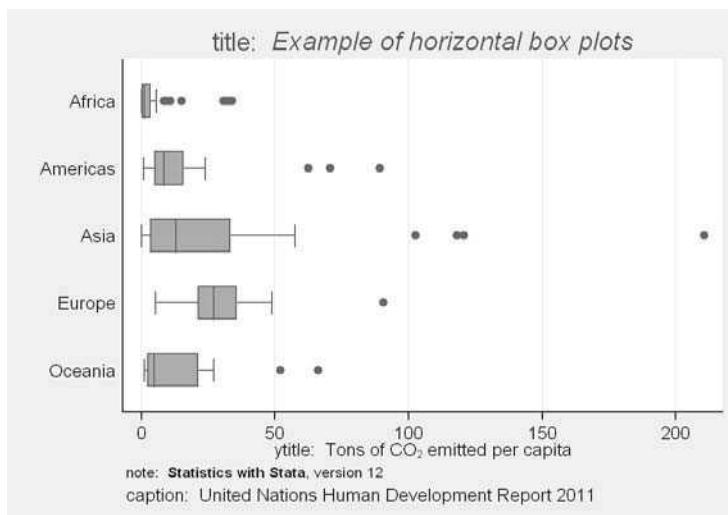


Figure 3.7

Individual outliers are not labeled in Figure 3.7 because they would be hard to read in the horizontal format. The three outliers in the Americas are the U.S., Canada, and Trinidad and Tobago (the leading Caribbean oil and gas producer). Australia has the highest per capita CO₂ in Oceania. Four oil-exporting nations comprise the very high outliers for Asia. Looking closer at outliers, which box plots make obvious, we often find that they are interesting observations in their own right and not just a statistical complication.

Numerous options control the appearance, shading and details of boxes in a box plot; type **help graph box** for a list. Axis labeling, tick marks, titles, and the **by(varname)** or **by(varname, total)** options work in a similar fashion with other Stata graphing commands. For example, **by(region)** would have drawn individual box plots in five small window panes, instead of five box plots in one graph as **over(region)** did in Figures 3.6 and 3.7.

Scatterplots and Overlays

Scatterplots belong to a broad family called **twoway** graphs. Stata's basic scatterplot command has the form

```
. graph twoway scatter y x
```

where *y* is the vertical or *y*-axis variable, and *x* is the horizontal or *x*-axis one. (The initial **graph** **twoway** part of this command is optional, but kept here to emphasize a family connection that becomes important later on.) For example, again using the *Nations2.dta* dataset, we could plot *life* (life expectancy) against *school* (mean years of schooling), with the result shown in Figure 3.8. Each point in Figure 3.8 represents one nation.

```
. graph twoway scatter life school
```

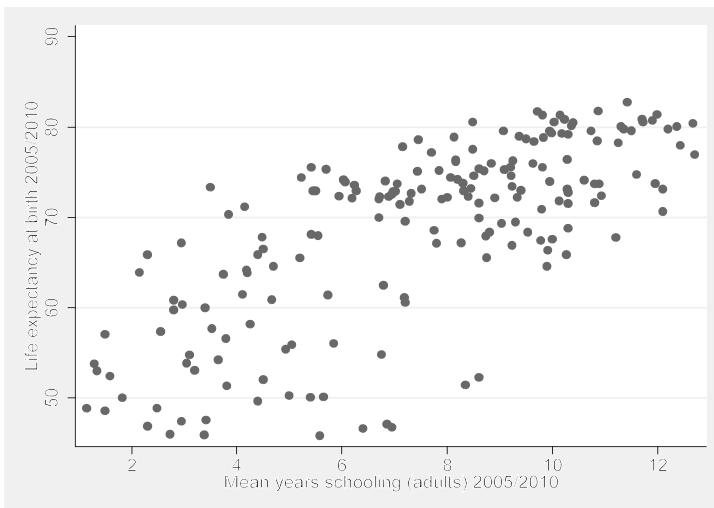


Figure 3.8

As with histograms, we can use **xlabel**, **xtitle**, **ylabel** and so forth to control x or y axis labels, titles etc. As with box plots, scatterplots also allow control of the shape, color, size, labeling and other attributes of markers. Figure 3.8 employs the default markers, which are solid circles. The same effect would result if we included the option **msymbol(O)**. Other possibilities include **msymbol(Th)** (large hollow triangles), **msymbol(d)** (small diamonds), **msymbol(plus)** (plus signs) or **msymbol(i)** (invisible symbols, handy for some purposes). Type **help scatter** for a complete list of markers and other options.

The **mcolor** option controls marker colors. For example, the command

```
. graph twoway scatter waste metro, msymbol(S) mcolor(purple)
```

would produce a scatterplot in which the symbols are large purple squares. Type **help colorstyle** for a list of available colors, which can also apply to bars, lines, text and other elements of any Stata graph.

One interesting possibility with scatterplots is to make symbol size (area) proportional to a third variable, thereby giving the data points different visual weight. For example, we might redraw the scatterplot of *life* against *school*, but make the symbol size reflect each country's population (*pop*). This is done in Figure 3.9, using the [**fweight=varname**] or frequency weight feature. Hollow circles, **msymbol(Oh)**, provide a suitable shape.

```
. graph twoway scatter life school [fweight=pop], msymbol(Oh)
```

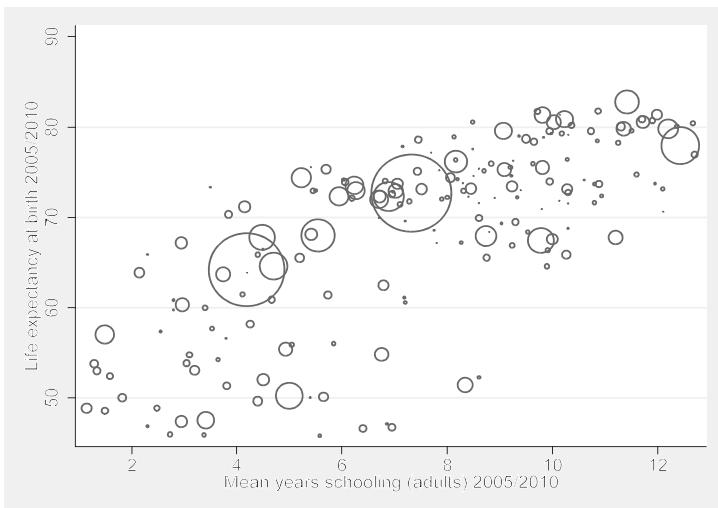


Figure 3.9

Frequency weights are useful with some other graph types as well. Weighting can be a deceptively complex topic because weights come in several types and have different meanings in different contexts. For an overview of weighting in Stata, type **help weight**.

A key feature of Stata's **graph twoway** family is that we can overlay two or more graphs to build more complex images. For example, to draw a scatterplot of *life* against *school*, with hollow circles as marker symbols, we would type

```
. graph twoway scatter life school, msymbol(Oh)
```

Simple regression lines (**lfit**) are a different **twoway** type. To see the line for *life* regressed on *school*, with a line of medium-thick width, type

```
. lfit life school, lwidth(medthick)
```

But often, we want to see the scatterplot and regression line together. That is accomplished by overlaying the **lfit** graph on top of the **scatter** graph using one command with **||** ("pipes") to indicate the overlay. The command below is shown as two lines, but it should be typed as one physical line.

```
. graph twoway scatter life school, msymbol(Oh)
    || lfit life school, lwidth(medthick)
```

Finally, if we have certain options that should apply to the image as a whole, these can be placed in the command after a final **||**. Figure 3.10 does this. The general options include not only **ylabel** and **xlabel** but also specify some details about the **legend**.

```
. graph twoway scatter life school, msymbol(Oh)
    || lfit life school, lwidth(medthick)
    || , ylabel(45(5)85) xlabel(2(2)12) xtick(1(2)13)
    legend(col(1) ring(0) position(11))
```

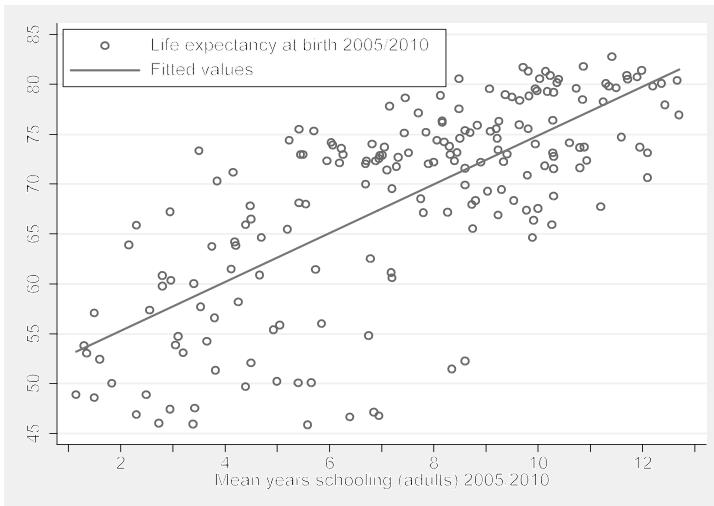


Figure 3.10

The **legend** option in Figure 3.10 specifies three things:

- col(1)** The legend should have just one column, and hence two rows.
- ring(0)** The legend is placed within the plot region, instead of outside it. A legend outside the plot region crowds the data into a smaller space.
- position(11)** The legend goes at the 11 o'clock position, which in this graph happens to be empty of data.

Placing a legend within the plot region but not over any actual data is a nice trick if we can manage it. By experimenting with defaults or other placements, you can see for yourself how this works. Consult **legend_options** under **help twoway options** for many more ways to control the position, contents and appearance of legends.

Figure 3.11 takes these ideas a step further in a graph with three overlays, one of them another **twoway** type: **lfitci**, meaning linear regression with confidence intervals. The **lfitci** graph is specified first, then two scatterplots are placed on top of that so we see their points over the gray confidence bands. If we specified **lfitci** last, the confidence bands would paint over some scatterplot points.

```
. graph twoway lfitci life school, lwidth(medthick)
    || scatter life school, msymbol(Th)
    || scatter life school if school > 8 & life < 55,
        msymbol(S) mlabel(country)
    || , ylabel(45(5)85) xlabel(2(2)12) xtick(1(2)13)
    legend(col(1) ring(0) position(11) label(3 "Life expectancy")
        order(3 2 1))
```

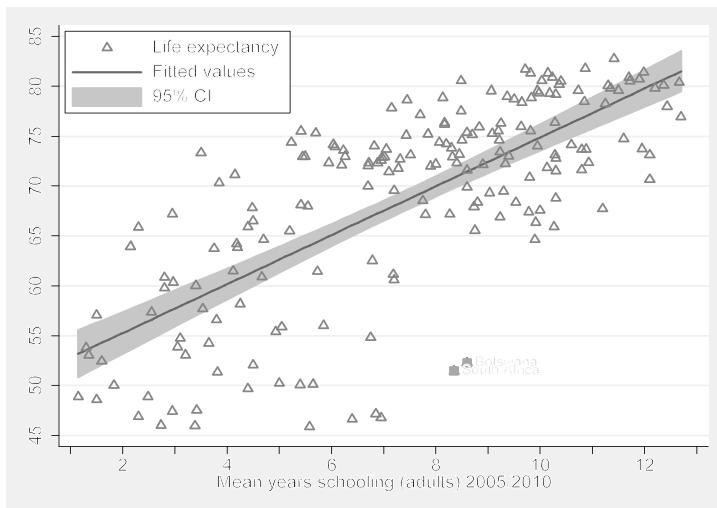


Figure 3.11

By default, **lfitci** shows confidence bands for the *conditional mean* of y , rather than for individual predicted values. Stata refers to standard errors for the conditional mean as “standard deviation of prediction” or **stdp**, so the default in Figure 3.11 is equivalent to typing

graph twoway lfitci life school, stdp

Standard errors for individual predicted values are termed “standard deviation of forecast” or **stdf**. To see the wider confidence bands for individual predictions, we could have typed instead
graph twoway lfitci life school, stdf

The other two plots in Figure 3.11 are both scatterplots, illustrating how to label (or plot with different symbols) certain selected observations. Identifying two outliers is accomplished here by drawing one ordinary scatterplot with all the data, and hollow triangles as markers:

|| scatter life school, msymbol(Th)

Then we overlay that with a second scatterplot (the third plot in this image) restricted by an **if** qualifier to countries for which mean schooling is greater than 8 years *and* life expectancy is greater than 55. Only two countries at lower right meet this criterion. They are plotted as solid squares (drawn over the triangles) and labeled with country names. Botswana and South Africa turn out to be the nations with this unusual combination of good education but poor life expectancy, which makes them stand apart from the up-to-right trend that characterizes most other nations.

**|| scatter life school if school > 8 & life < 55,
msymbol(S) mlabel(country)**

The overall options for Figure 3.11 specify x axis labels and tick marks, and also control the legend. Again we give the legend one column, and place it at 11 o’clock within the plot region. The label for the third y variable in the legend is specified as “Life expectancy” instead of the much longer variable label that would be used by default.

legend(col(1) ring(0) position(11) label(3 "Life expectancy"))

The **legend()** option ends with an **order(3 2 1)** suboption specifying that we want the legend items to be in order 3-2-1. This is not quite as simple as it appears because to Stata's way of thinking our three overlaid plots in Figure 3.11 actually involve four variables that could be in the legend. Given numbers by their sequence in the initial **graph twoway** command, these are (1) the 95% confidence interval, (2) the fitted values or linear regression predictions, (3) life expectancy for the full dataset, our first overlaid scatterplot, and (4) life expectancy for just Botswana and South Africa, our second overlaid scatterplot. By specifying **order(3 2 1)** we asked for the legend in Figure 3.11 which lists (3) first, (2) second and (1) last — and leaves (4) out, because it is not mentioned in the **order()** suboption. Thus, the **order()** suboption controls not only what order variables appear in the legend, but whether they appear at all.

Scatterplot matrices are not **twoway** plot types and cannot be overlaid with other graphs, but they involve multiple scatterplots that follow the same marker symbol conventions. Figure 3.12 shows a scatterplot matrix for five variables from *Nations2.dta*.

```
. graph matrix gdp school adfert chldmort life, half msymbol(oh)
```

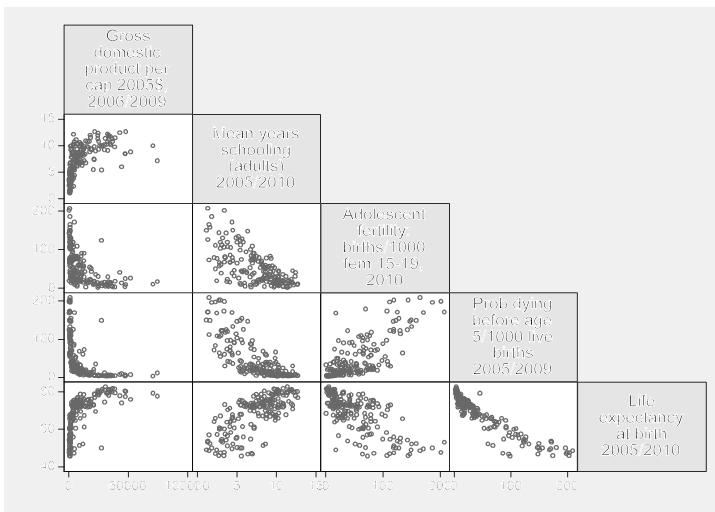


Figure 3.12

A scatterplot matrix is the visual counterpart to a correlation matrix, which can prove useful in multivariate analysis. They provide a compact display of relationships between a number of variable pairs, allowing the analyst to scan for signs of nonlinearity, outliers or clustering that might affect statistical modeling. Nonlinear relationships involving *gdp* (per capita Gross Domestic Product) stand out prominently in Figure 3.12, giving a warning we would not see from the correlation matrix alone.

The **half** option specified that Figure 3.12 should include only the lower triangular part of the matrix. The upper triangular part is symmetrical and basically redundant. **msymbol(o)** called for small circles as markers, just as we might with a scatterplot. Control of the axes is more complicated, because there are as many axes as variables; type **help graph matrix** for details.

When the variables of interest include one dependent or effect variable, and several independent or cause variables, it helps to list the dependent variable last in the **graph matrix** command's variable list. That results in a neat row of dependent-versus-independent variable (*y* vs. *x*) graphs across the bottom row of the matrix. The last-named or bottom-row variable in Figure 3.12, *life* (life expectancy), is analyzed as a dependent variable in Chapters 7 and 8.

Line Plots and Connected-Line Plots

Mechanically, connected-line plots (**graph twoway connect**) are just scatterplots in which the points are connected by line segments. Line plots (**graph twoway line**) show the line segments without markers for the scatterplot points. Both belong to Stata's versatile **graph twoway** family, which can be overlaid in any combinations. The scatterplot options that control axis labeling and markers work much the same with connected-line and line plots as well. Further options control width, pattern, color and other characteristics of the lines themselves.

Connected-line and line plots tend to have different uses than scatterplots. For example, they serve well to graph changes in a variable over time. To illustrate, we return to dataset *Arctic9.dta*, which contains 33 years of Arctic sea ice and temperature observations.

```
. use C:\data\Arctic9.dta, clear
. describe

Contains data from C:\data\Arctic9.dta
obs: 33
vars: 8
size: 891
Arctic September mean sea ice
 1979-2011
 17 Apr 2012 09:21

variable   storage      display      value
          name       type        format    label
year        int         %ty
month       byte        %8.0g
extent      float        %9.0g
area        float        %9.0g
volume      float        %8.0g
volumehi   float        %9.0g
volumelo   float        %9.0g
tempN      float        %9.0g
                                variable label
                                Year
                                Month
                                Sea ice extent, million km^2
                                Sea ice area, million km^2
                                Sea ice volume, 1000 km^3
                                Volume + 1.35 (uncertainty)
                                Volume - 1.35 (uncertainty)
                                Annual air temp anomaly 64N-90N C

Sorted by: year
```

A line plot of *area* against *year* depicts the reduction in September sea ice area over these years, and makes an abrupt drop in 2007 particularly apparent (Figure 3.13).

```
. graph twoway line area year,
    title("Arctic sea ice, September 1979-2011")
```

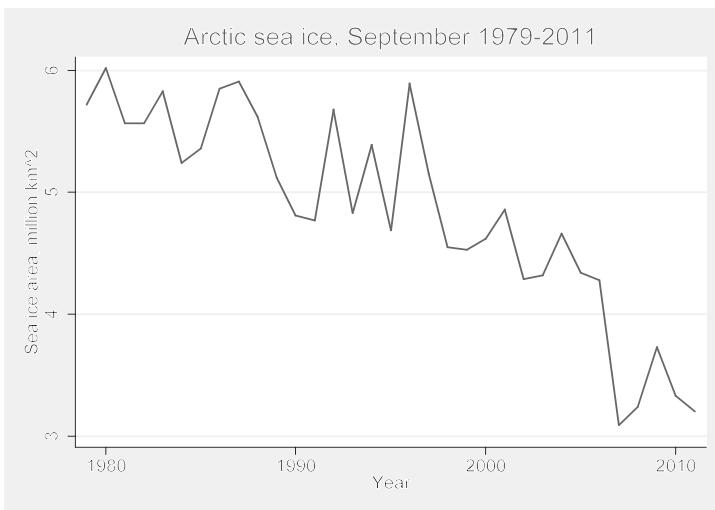
**Figure 3.13**

Figure 3.14 offers a more elaborate time plot, adding a second line for sea ice *extent* (the area covered by at least 15% ice). We label the *x* axis from 1980 to 2010 in 5-year increments (`xlabel(1980(5)2010)`) but suppress the *x*-axis title because “Year” is obvious, and unnecessarily wastes data space. The *y* axis extends down to zero, a possible state of some interest to Arctic observers. *Yaxis labels* go from 0 to 8 in increments of 1, with grid lines including the minimum (0) and maximum (8) labeled values: `ylabel(0(1)8, grid gmin gmax)`.

```
. graph twoway line area extent year, xlabel(1980(5)2010) xtitle("")
    lwidth(medium medthick) lpattern(solid dash)
    legend(row(2) ring(0) position(9)
        label(1 "Area") label(2 "Extent") order(2 1))
    ylabel(0(1)8, grid gmin gmax) ytitle("Million km{superscript:2}")
    title("Arctic sea ice, September 1979`=char(150)'2011")
```

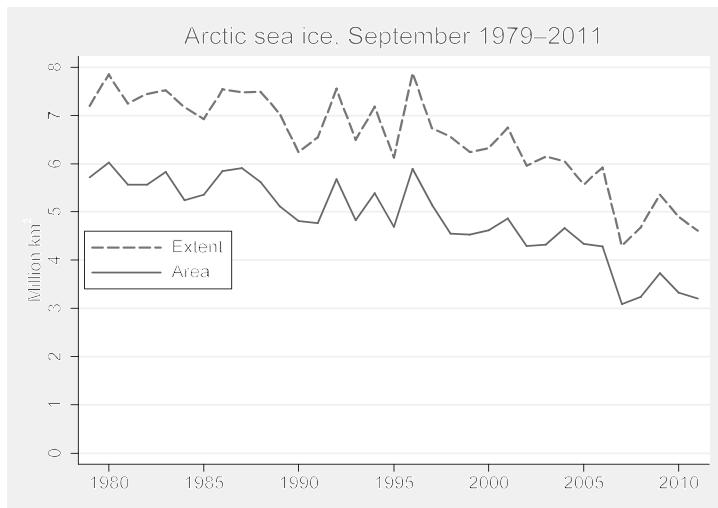


Figure 3.14

The first-named *y* variable in the Figure 3.14 command (*area*) is graphed with a line of **medium** width and **solid** pattern. The second *y* variable (*extent*) is graphed with a heavier line of medium-thick (**medthick**) width and **dashed** pattern. **legend()** options simplify the labels in the legend, and place *extent* first corresponding to its higher position in the graph.

Ice area and extent are measured in millions of square kilometers. The **ytitle("Million km^{superscript:2}")** option displays this as "Million km²". Type **help graph text** for more about controlling text attributes in graphs. Figure 3.7 gave some other examples.

Another typographical trick in Figure 3.14 is more subtle: the years 1979–2011 in the title are separated by an n-dash (–) rather than a hyphen (-). The n-dash is a standard character that does not appear on your keyboard, but is identified by the ASCII code 150. The **title** option for Figure 3.14 inserted ASCII character 150 between 1979 and 2011 by writing **1979'=char(150)'2011**. Note the use of left and right single quotes.

Figure 3.15 employs the n-dash along with two other ASCII characters in a time plot where the two *y* variables have different scales and are graphed together using overlays. For this example we overlay a line plot of *area* vs. *year* with a connected-line plot (**connect**, which combines the features of **scatter** with **line**) of Arctic temperature vs. *year*. The graph shows September sea ice declining as annual Arctic air temperatures rise; the two variables influence each other, and both reflect larger changes originating outside of the Arctic.

```
. graph twoway line area year, ylabel(0(1)6) yline(0)
    ytitle("Ice area, million km`=char(178)'")
    || connect tempN year, yaxis(2) ylabel(-1(1)2,
        axis(2)) msymbol(+)
    ytitle("Arctic temperature anomaly, `=char(186)'C", axis(2))
    || , xlabel(1980(5)2010) xtitle("")
```

```

legend(row(2) ring(0) position(6)
      label(1 "Ice area") label(2 "Temperature") order(1 2))
title("Arctic September sea ice and annual temperature,
      1979`=char(150)'2011", size(medium))

```

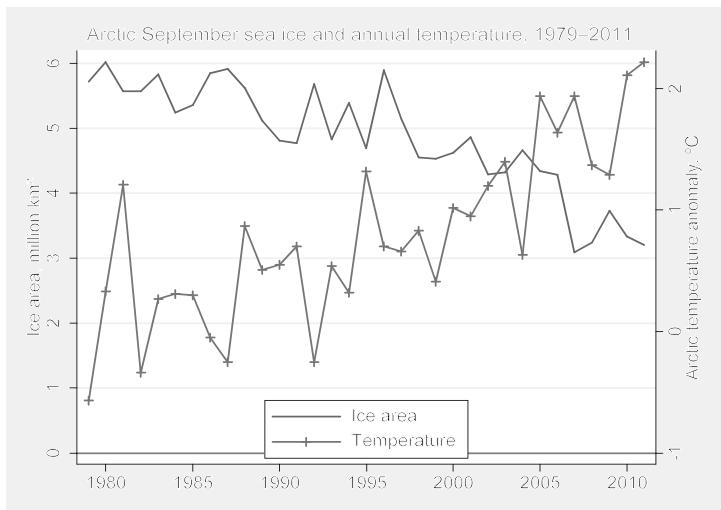


Figure 3.15

In Figure 3.15, Arctic temperature anomaly is assigned to **yaxis(2)**, which is the right-hand *y* axis. Temperature values are marked as plus signs (**msymbol(+)**). A legend inside the data region at 6 o'clock position identifies the two variables. Instead of using **{superscript:2}** to write km^2 on the left-hand *y* axis (as was done earlier in Figure 3.14), Figure 3.15 uses ASCII character 178—which also is a superscript 2 but looks slightly better. On the right-hand *y* axis, ASCII character 186 provides the degree symbol for $^{\circ}\text{C}$.

How do we know that character 150 is an n-dash, character 186 a degree symbol, and so forth? Figure 3.16 gives a table of ASCII codes, drawn by a convenient utility named **asciiplot**. This is not part of Stata but written in Stata's programming language by geographical statistician Nicholas Cox. Type **findit asciiplot** for instructions on how to download and install this program, which makes the table in Figure 3.16 available whenever you type **asciiplot**.

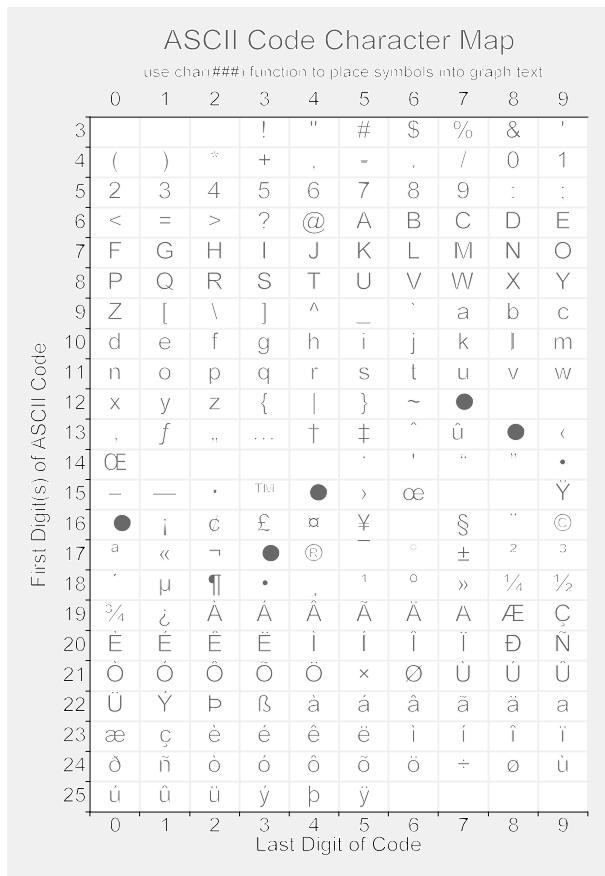


Figure 3.16

Figure 3.15 illustrated the simplest **connect** style, in which data points are connected by straight line segments. Other **connect** choices are listed below. The default straight lines correspond to **connect(direct)** or **connect(l)**. For more details, see **help connectstyle**.

connect()	Abbreviation	Description
none	i	do not connect
direct	l (letter “el”)	connect with straight lines
ascending	L	direct, but only if $x[i+1] > x[i]$
stairstep	J	horizontal, then vertical
stepstair	(none)	vertical, then horizontal

Other Twoway Plot Types

In addition to basic line plots and scatterplots, the **graph twoway** command can draw a wide variety of other types. This section illustrates several more; type **help graph twoway** for a complete list.

Earlier, in Figures 3.10 and 3.11, we used **graph twoway lfit** (linear fit) to draw a simple regression line. A similar command, **graph twoway qfit**, draws a quadratic or second-order polynomial regression curve. Figure 3.17 illustrates using another variable from the *Arctic9.dta* dataset, sea ice *volume*. Volume is measured in cubic kilometers, so ASCII character 179 here provides the superscript for km³. Writing {superscript:3} would also have worked.

```
. graph twoway connect volume year, ylabel(0(2)16) yline(0)
    ytitle("Volume, thousand km`=char(179)'")
    || qfit volume year, lwidth(medthick)
    || , xlabel(1980(5)2010) xtitle("")
    title("Arctic sea ice, September 1979`=char(150)'2011")
    legend(row(2) ring(0) position(6)
    label(1 "Volume") label(2 "quadratic fit") order(1 2))
```

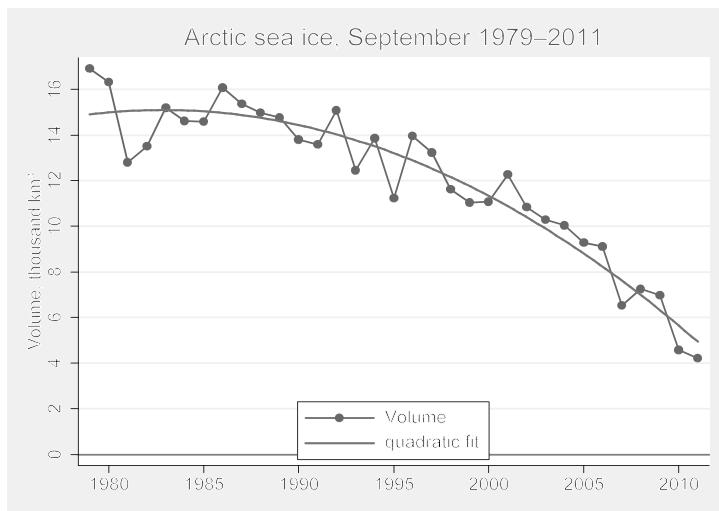


Figure 3.17

The quadratic curve together with a horizontal line drawn at $y = 0$ (**yline(0)**) emphasize that mean September ice volume has been falling toward the floor. A linear model would not capture its accelerating trajectory, but in some respects the quadratic mode is unrealistic too. The curve in Figure 3.17 shows a slight but counterfactual slight rise in the early years, and if extrapolated to future years it would predict ice volumes below zero. In Chapter 8 we return to these data and consider a more plausible curve.

Figure 3.18 takes a different look at the same *volume* time series, this time using a range-area plot, **graph twoway rarea**, to show 95% uncertainty limits of plus or minus 1.35 thousand km³

(Schweiger et al. 2011). The range-area or **rarea** command looks for two *y* variables — in this example, *volumehi* and *volumelo* — that define the upper and lower bounds of an area to be shaded. A line plot of *volume* itself is drawn with a thick line (**lwidth(thick)**) and overlaid on top of the **rarea** plot so it can be seen over the shading.

```
. graph twoway rarea volumehi volumelo year, color(gs13)
    || line volume year, lwidth(thick) lcolor(green)
    || , ylabel(0(2)18) yline(0)
    ytitle("Volume, thousand km` =char(179)'")
    title("Arctic sea ice, September 1979` =char(150)'2011")
    legend(row(2) ring(0) position(7))
    label(1 "uncertainty") label(2 "Volume") order(2 1))
```

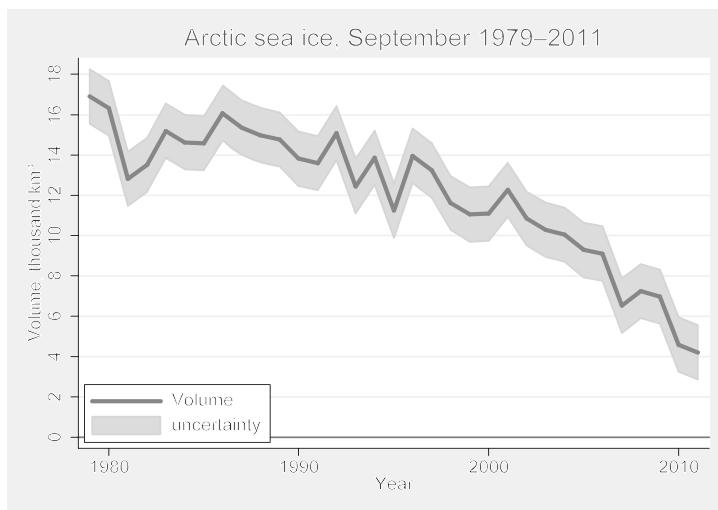
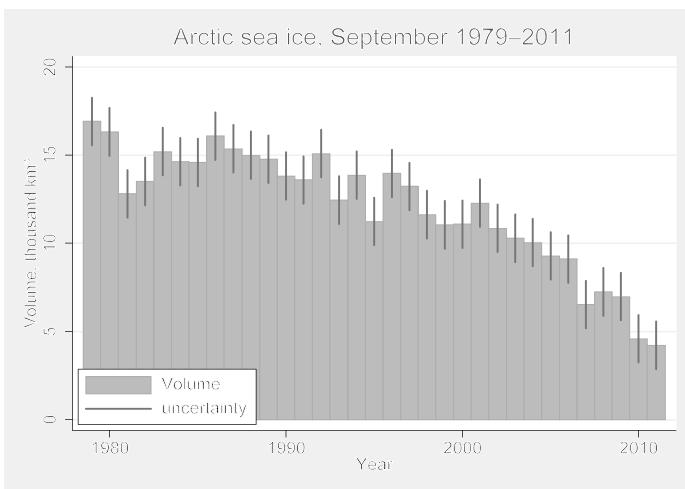


Figure 3.18

The **graph** command drawing Figure 3.18 also specifies colors for both the **rarea** and **line** plots. **color(gs13)** assigns the **rarea** bands a gray scale color of 13, closer to white (**gs16**) than to black (**gs0**). The **line** plot overlaid on these bands is colored **green**. Controlling the colors (instead of going with their defaults) does little for an image published in black and white, but has obvious value for other media. Type **help colorstyle** to see a list of colors available for the elements of Stata graphics.

Figure 3.19 overlays two other **graph twoway** types to make a different image from the same Arctic ice volume information. In this image the *volume* estimates themselves are drawn by **graph twoway bar**, while the uncertainty bands appear as range spikes (**rspike**) instead of the range area (**rarea**) we saw in Figure 3.18. The bars have a medium gray color (**color(gs10)**). Note that Figure 3.18 overlays the *volume* line on top of the uncertainty range area, whereas Figure 3.19 overlays the uncertainty range spikes on top of the *volume* bars. The order in both cases is required for visibility. You can experiment with similar commands to see how that works.

```
. graph twoway bar volume year, color(gs10)
    || rspike volumehi volumelo year, lwidth(medthick)
    || , ytitle("Volume, thousand km`=char(179)'")
    title("Arctic sea ice, September 1979`=char(150)'2011")
    legend(row(2) ring(0) position(7)
        label(1 "Volume") label(2 "uncertainty"))
```

**Figure 3.19**

graph twoway bar should not be confused with **graph bar**, a different command introduced in the next section. **twoway bar** is basically a *y*-versus-*x* plot that assumes two measurement variables, and shares features with other **twoway** types such as *y* and *x* axis labeling and the possibility of overlays.

Stata offers more than 40 different **graph twoway** types, too many to catalog here. Later chapters demonstrate several others; type **help graph twoway** for the complete list, with links to the syntax of individual commands.

Bar Charts and Pie Charts

The **graph bar** command, unlike **graph twoway bar**, works well to display relationships involving one or more categorical variables. Such graphs prove particularly useful with survey data, as will be shown in Chapter 4. This section serves just to introduce the command, with an example using variables from the cross-national dataset *Nations_2.dta*.

```
. use C:\data\Nations2.dta, clear
. describe region gdp pop
```

variable name	storage type	display format	value label	variable label
region	byte	%8.0g	region	Region
gdp	float	%9.0g		Gross domestic product per cap 2005\$, 2006/2009
pop	float	%9.0g		Population 2005/2010

Variable *gdp*, per capita Gross Domestic Product, has values ranging from 279.8 to 74,906 dollars per person. Five-digit numbers often look unwieldy as axis labels, so we start by generating a new variable, *gdp1000*, expressing Gross Domestic Product in thousands of dollars. Figure 3.20 depicts the mean and median of *gdp1000* over values of *region*, using **graph bar**.

```
. generate gdp1000 = gdp/1000
. summarize gdp gdp1000

      variable |   Obs        Mean    Std. Dev.       Min       Max
                 | 179  12118.74  13942.34  279.8  74906
      gdp1000 | 179  12.11874  13.94234  .2798  74.906

. graph bar (mean) gdp1000 (median) gdp1000, over(region)
    ytitle("Per capita GDP, thousands of 2005 US dollars")
    xlabel(bar, format(%3.1f))
    bar(1, color(blue)) bar(2, color(orange))
    legend(ring(0) position(11) col(2) label(1 "Mean")
          label(2 "Median") symxsize(*.5))
```

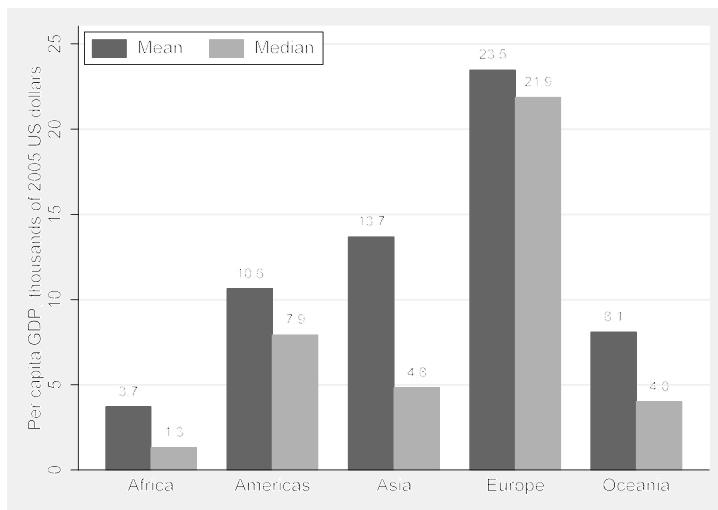


Figure 3.20

Figure 3.20 includes labels giving the height of each bar, displayed in numerical **format(%3.1f)** — which means a fixed format with three digits, one of them right of the decimal. **graph bar** can display not only means or medians but also other statistics including various percentiles, minimum, maximum or count. It also could show these statistics for more than one variable, if they have similar scales.

The legend in Figure 3.20 is placed within the data region at 11 o'clock: **ring(0) position(11)**. It has two columns to parallel the side-by-side arrangement of bars. **symxsize(*.5)** causes the horizontal (*x*) dimension of the symbols in the legend to appear at half their default size, which saves space and also makes the symbols more similar to the widths of the bars themselves.

The **graph bar** example above specifies a blue color for bar 1 and orange for bar 2. Blue and orange may not show on this page, but even converted to black and white the blue and orange bars look quite different. As Nicholas Cox has pointed out, blue and orange form a noteworthy pair because they appear visually distinct to readers with most types of color vision deficiencies, unlike (for example) red and green. Analysts might take this consideration into account when designing graphs where distinguishing between colors is critical.

Bar charts can provide clear visualizations of relationships involving many categories and two or more variables. Pie charts, on the other hand, rarely clarify the analysis but are popular for some public presentations. Figure 3.21 illustrates Stata's **graph pie** command, showing the breakdown of world population by region. The variable *pop* ranges from just under 10,000 to 1.32 billion (1.32e+09, meaning $1.32 \times 10^9 = 1,320,000,000$). To make our pie chart readable it helps to create a new variable, *popmil* or population in millions.

```
. gen popmil = pop/1000000
. summarize pop popmil

Variable |   Obs      Mean    Std. Dev.      Min      Max
-----+-----+-----+-----+-----+-----+
pop |   194  3.44e+07  1.31e+08    9767  1.32e+09
popmil |   194  34.37752 131.4004   .009767 1324.696

. graph pie popmil, over(region) pie(2, explode)
    plabel(_all sum, format(%4.0f))
    title("World population in millions, by region")
    legend(col(1) position(9))
```

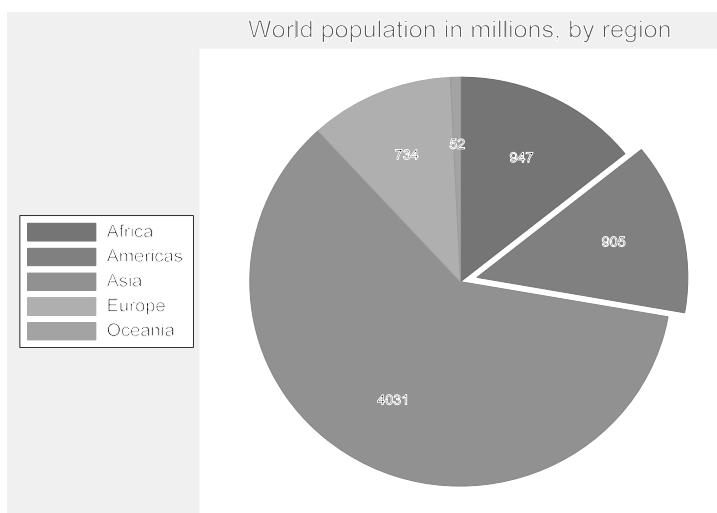


Figure 3.21

The **pie(2, explode)** option “explodes” the second pie slice (Americas) for emphasis. **plabel(_all sum, format(%4.0f)** requests labels for all of the pie slices, giving the sum of *popmil* (that is, the total population in millions) for each region. The pie labels are have **format(%4.0f)**, meaning a fixed numerical format with four digits, none right of the decimal.

Type **help graph pie** to see other pie chart options, including methods for differently-organized data. One interesting variation uses the **by()** option to create an image containing multiple small pie charts that can be visually compared.

Symmetry and Quantile Plots

Box plots, bar charts and histograms summarize measurement variable distributions, hiding individual data points to clarify overall patterns. Symmetry and quantile plots, on the other hand, include points for every observation. They take more effort to read than summary graphs, because they convey more detailed information.

A histogram of the ratio of females to males in the labor forces of 177 countries, from *Nations2.dta*, appears in Figure 3.22. A superimposed normal (Gaussian) curve indicates that *femlab* has a heavier-than-normal left tail (countries with relatively few females in the labor force), and a lighter-than-normal right tail — the definition of negative skew.

```
. histogram femlab, norm percent
```

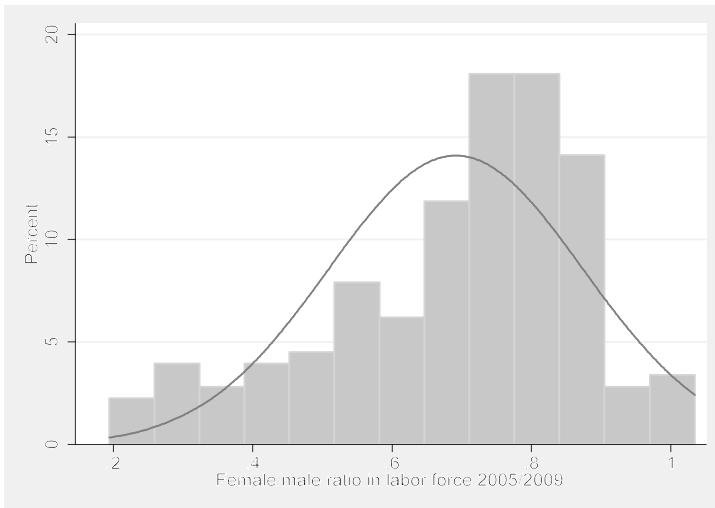


Figure 3.22

Figure 3.23 depicts this distribution as a symmetry plot. It plots the distance of the *i*th observation above the median (vertical) against the distance of the *i*th observation below the median. All points would lie on the diagonal line if this distribution were symmetrical. Instead,

we see that distances below the median grow steadily larger than corresponding distances above the median, a symptom of negative skew.

```
. symplot femlab
```

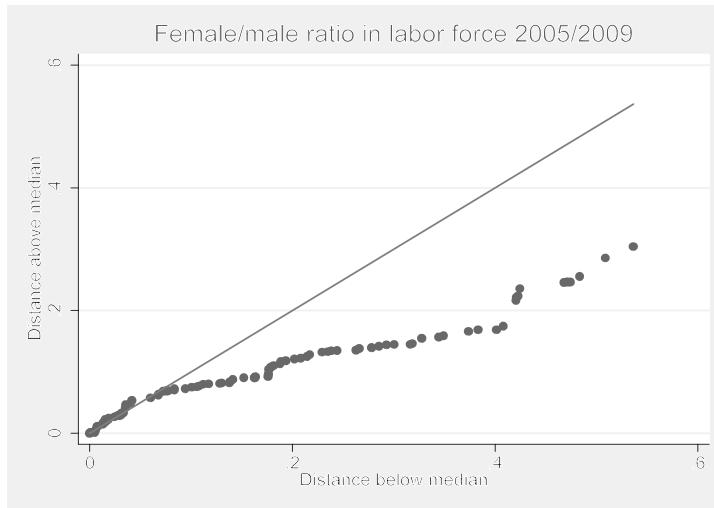


Figure 3.23

Quantiles are values below which a certain fraction of the data lie. For example, a .3 quantile is that value higher than 30% of the data (similar to the 30th percentile). If we sort n observations in ascending order, the i th value forms the $(i-.5)/n$ quantile. Quantile plots automatically calculate what fraction of the observations lie below each data value, and display the results graphically as in Figure 3.24. Quantile plots provide a reference for someone who does not have the original data at hand. From well-labeled quantile plots, we can estimate order statistics such as median (.5 quantile) or deciles (.1, .2, .3 quantiles and so forth). We could also read a quantile plot to estimate the fraction of observations falling below a given value.

```
. quantile femlab, xlabel(0(.1)1, grid) ylabel(.2(.1)1, grid)
```

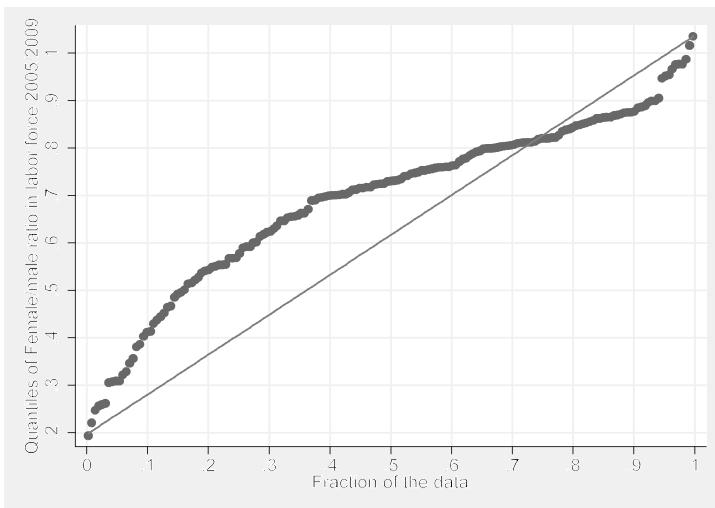


Figure 3.24

Quantile–normal plots, also called normal probability plots, compare quantiles of a variable’s distribution with quantiles of a theoretical normal distribution having the same mean and standard deviation. They allow visual inspection for departures from normality in every part of a distribution, which can help guide decisions regarding normality assumptions and efforts to find a normalizing transformation. Figure 3.25, a quantile–normal plot of *femlab*, confirms the negative skew noticed earlier. The **grid** option calls for a set of lines marking the .05, .10, .25 (first quartile), .50 (median), .75 (third quartile), .90, and .95 quantiles of both distributions. The .05, .50, and .95 quantile values are displayed along the top and right-hand axes.

```
. qnorm femlab, grid
```

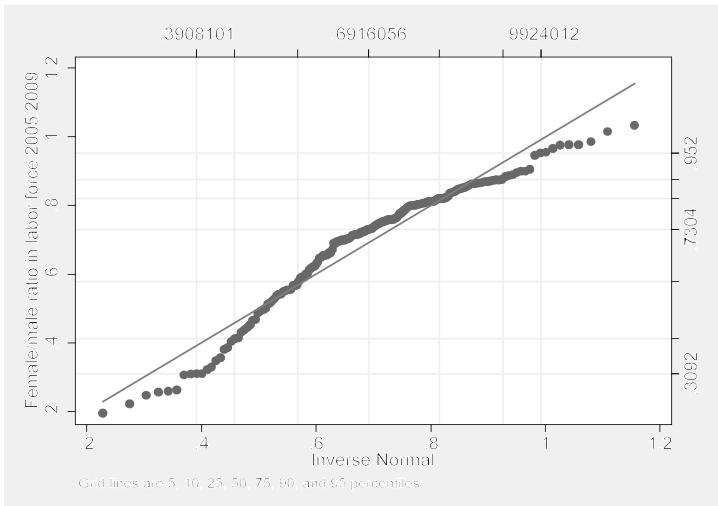


Figure 3.25

Quantile–quantile plots (not shown) resemble quantile–normal plots, but compare quantiles (ordered data points) of two empirical distributions instead of comparing one empirical distribution with a theoretical normal distribution. *Regression with Graphics* (Hamilton 1992a) includes an introduction to reading these and other quantile-based plots. Chambers et al. (1983) provide more details. Related Stata commands include **pnorm** (standard normal probability plot), **pchi** (chi-squared probability plot) and **qchi** (quantile–chi-squared plot). Type **help quantile** for details on this graphical family.

Adding Text to Graphs

Titles, captions and notes can be added to make graphs more self-explanatory. The default versions of titles and subtitles appear above the data region; notes (which might document the data source, for instance) and captions appear below. See Figure 3.7 for an example using title, caption and note. These defaults can be overridden, of course; type **help title options** for more information about placement of titles, or **help textbox options** for details concerning their content, including font size and color.

It also is possible to add text boxes at specified locations within the data region. For example, we might wish to annotate the plot of a time series, such as the historical ice-out dates on Lake Winnepeaukee, as done in Figure 3.26.

```
. use C:\data\lakesunwin.dta, clear
. describe
```

```

Contains data from C:\data\lakesunwin.dta
obs: 144
vars: 5
size: 1,728

Sunapee & Winnipesaukee ice out
1869-2012
27 Mar 2012 09:46

variable   storage   display   value
name      type      format    label
year       int       %ty
sunedate   float     %tdcymd
sunout     int       %9.0g
winedate   int       %tdcymd
winout     int       %9.0g

Sorted by: year

. graph twoway line winout year
|| lowess winout year, lwidth(medthick)
|| if year>=1887 , xlabel(1890(10)2010) legend(off) xtitle("")
ytitle("Ice Out day of year")
text(87 1905 "Early 20th century" "warming")
text(130 1950 "Mid-century cooling" "(global dimming)",
    box margin(small) bcolor(gs11))
text(125 1998 "Modern rapid" "warming", justification(left))

```

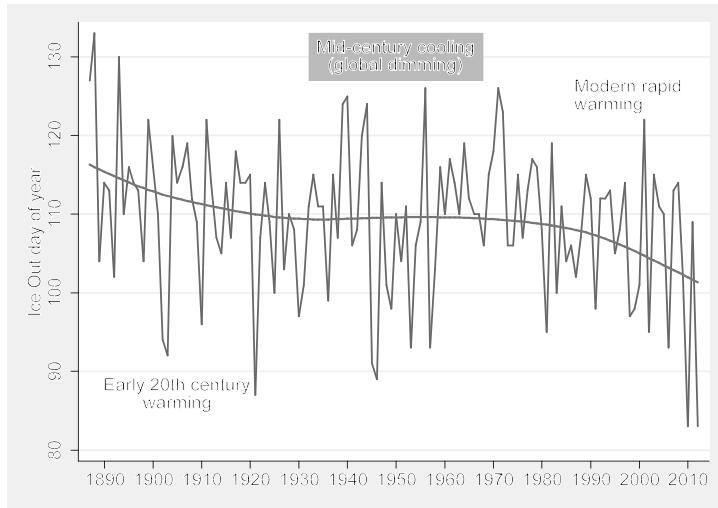


Figure 3.26

The basic graph in Figure 3.26 is a **twoway line** plot of Winnepeaukee ice-out day against years from 1887 through 2012. Overlaid on this is a second twoway plot showing a smooth **lowess** regression curve, drawn with a medium-thick line (**lwidth(medthick)**) to stand out. Lowess regression, explained in Chapter 8, provides a flexible approach to data smoothing that has several advantages over the better-known moving-average methods. The lowess curve in Figure 3.26 reveals multi-decade trends that underlie the jagged year-to-year variations.

Decadal trends on this lake broadly follow patterns of New Hampshire and global temperature over the same years (Hamilton et al. 2010a). Ice-out dates generally declined during a warming period of the late 19th and early 20th century. Slight cooling in mid-century, from the 1940s to the early 1970s, resulted in later ice-out dates. Globally, this period marks a time when sunlight reaching the surface was measurably dimmed by industrial pollution (Wild et al. 2007). During the warming from the mid-1970s on, Winnepeaukee ice-out dates more rapidly declined. The earliest recorded ice-out dates occurred in 2010 and 2012.

Three text boxes within the data region in Figure 3.26 note these climate episodes. They each contain two lines of text, broken within the `text()` options by separate sets of double quotes. The `text()` options specify *y* and *x* coordinates for the boxes; it often takes some experimenting to place them exactly where you want. The first box,

```
text(87 1905 "Early 20th century" "warming")
```

is centered at *y* = 87 and *x* = 1905, with default attributes. The second box, at *y* = 130 and *x* = 1950, is shown surrounded by a visible border with `small` margin from the text. Border and background colors are specified as a medium gray, `gs11`.

```
text(130 1950 "Mid-century cooling" "(global dimming)",  
    box margin(small) bcolor(gs11))
```

The third box contains left-justified text.

```
text(125 1998 "Modern rapid" "warming", justification(left))
```

See `help textbox options` and `help colorstyle` for more on controlling how text boxes look.

Graphing with Do-Files

Complicated graphics like Figure 3.26 require `graph` commands that are many physical lines long (although Stata views the whole command as one logical line). Do-files, introduced in Chapter 2, help in writing such multi-line commands. They also make it easy to save the command for future re-use, in case we later want to modify the graph or draw it again.

The following commands, typed into Stata's Do-file Editor and saved with the file name `fig03_26.do`, become a new do-file for drawing Figure 3.26.

```
use C:\data\lakesunwin.dta, clear  
graph twoway line winout year ///  
    || lowess winout year, lwidth(medthick) ///  
    || if year>=1887 , xlabel(1890(10)2010) legend(off) xtitle("") ///  
    ytitle("Ice Out day of year") ///  
    text(87 1905 "Early 20th century" "warming") ///  
    text(130 1950 "Mid-century cooling" "(global dimming)", ///  
        box margin(small) bcolor(gs11)) ///  
    text(125 1998 "Modern rapid" "warming", ///  
        justification(left))  
graph save Graph C:\graphs\fig03_26.gph, replace  
graph export C:\graphs\fig03_26.png, as(png) replace  
graph export C:\graphs\fig03_26.eps, as(eps) replace
```

Ending lines with `///` in this do-file tells Stata that the command continues on the next physical line. The command executes only after it reaches a line that does not end with `///`. An alternative

way to write multiline commands is to give a **#delimit ;** command, which would set the end-of-line delimiter to a semicolon instead of the default Enter or carriage return (**#delimit cr**), as illustrated in Chapter 2.

The **graph save Graph** command saves the image (by default, temporarily named “Graph” as seen in the Graph window) in Stata’s .gph format. We can always specify our own temporary name for a graph, by adding to the **graph** command an option such as **name(newgraph)** or **name(fig03_26)**. Such temporary names for graphs become important when we have several graphs currently displayed, and want to save or print a particular one. Giving a temporary name to the graph as we create it does *not* save that graph to disk. The temporary and saved-file names need not be the same. Type **help name option** for more discussion.

Our do-file’s **graph export** commands create second and third versions of the same image in different formats. Portable Network Graphics (.png) files are bitmapped images and have a fixed resolution, but they are very compatible and easily shared through Web pages, Powerpoint presentations or other applications. Encapsulated PostScript (.eps) is a high-quality vector format often preferred for publication. Type **help graph export** to learn about other options for this command.

Once the graph-drawing and graph-saving commands are saved as *fig03_26.do*, simply typing the command

```
. do fig03_26
```

causes this do-file to execute, redrawing the graph and saving it in three formats, writing over earlier versions of those files if they exist.

Retrieving and Combining Graphs

Any graph saved in Stata’s “live” .gph format can subsequently be retrieved into memory by the **graph use** command. For example, we could retrieve Figure 3.26 by typing

```
. graph use fig03_26.gph
```

Once the graph is in memory, it is displayed onscreen and can be printed or saved again with a different name or format. From a graph saved earlier in .gph format, we could subsequently save versions in other formats such as Encapsulated PostScript (.eps), Portable Network Graphics (.png) or Enhanced Windows metafile (.emf). It is also possible to change the color scheme, either through menus or directly in the **graph use** command. *fig03_26.gph* was saved in Stata’s s2color scheme, but we could see how it looks in s2monochrome (which substitutes dashed lines for different colors) by typing

```
. graph use fig03_45.gph, scheme(s2mono)
```

Graphs saved on disk can also be combined by the **graph combine** command. This provides a way to bring multiple plots into the same image.

The do-file below (saved as *fig03_27.do*, then run by typing **do fig03_27**) combines Figures 3.17, 3.18 and 3.19 from this chapter. The **///** indicates a command continued on the next physical line, as described earlier. The final combined image is saved as Figure 3.27.

```
graph combine ///
C:\A_books\SwS_12\graphs\fig03_17.gph ///
C:\A_books\SwS_12\graphs\fig03_18.gph ///
C:\A_books\SwS_12\graphs\fig03_19.gph ///
, rows(2) altshrink ///
title("Combining Figures 3.17-19", size(medium))
graph save Graph C:\graphs\fig03_27.gph, replace
graph export C:\graphs\fig03_27.emf, as(emf) replace
```

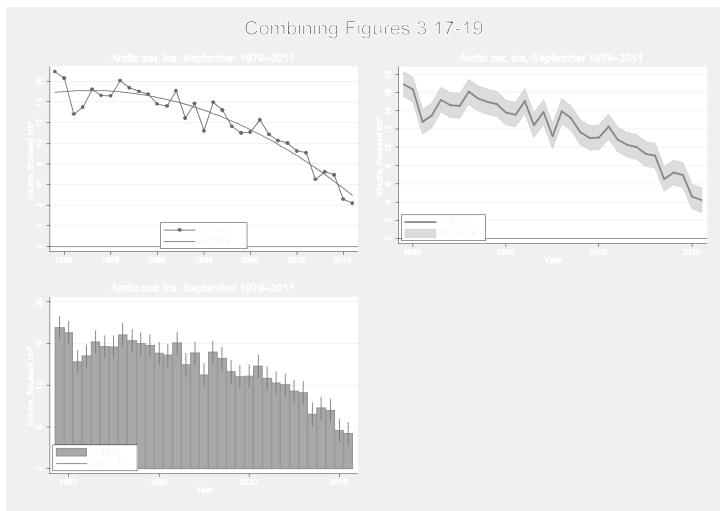


Figure 3.27

The **rows(2)** option specified that Figure 3.27 should be arranged with the sub-graphs in two rows. Equivalently, we could have specified **col(2)**. **altshrink** calls for alternate scaling of the text within each small image. Note that we can request an overall **title** (or **note**, **caption**, **ytitle**, **xtitle** and so forth — not shown) for the combined image. However, we cannot substantially change contents of the sub-graphs themselves.

Graph Editor

The Graph Editor allows us to alter the appearance of a graph currently in memory, whether just drawn or previously saved and retrieved by **graph use**. It is easier to learn about this useful feature by experimenting yourself, rather than reading from a book. As an example to get started, however, we can show how to make a few changes in Figure 3.18. A first step is to retrieve this graph into memory.

```
. graph use C:\graphs\fig03_18.gph
```

Then, in the Graph window, select File > Start Graph Editor, or click the Graph Editor icon . The view will change to include a Tools Toolbar on the left margin and an Object Browser on the right, as seen in Figure 3.28. The Tools Toolbar contains a pointer tool for selecting parts of the graph, and other tools to add text, add lines, add markers, and edit the graph grid. The Object Browser presents a hierarchical list of graph contents. Some items are marked with a + sign, and clicking on this + will expand the list to show lower objects within it. We can select objects in the graph by clicking the pointer on their image, or by clicking names in the Object Browser (often easier in complex graphs).

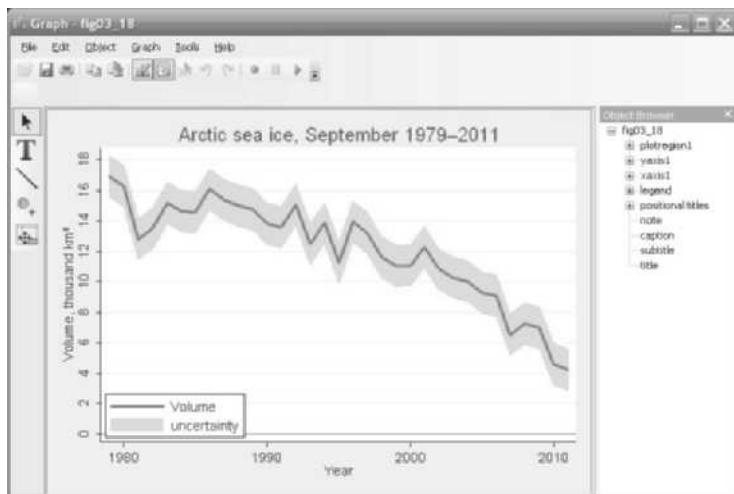


Figure 3.28

“Plot1” in this image refers to the **twoway rarea** plot forming the gray error bands. Click on the gray band (or plot1 under plotregion1 at right) to select them. Plot1 is now highlighted both in the Object Browser, and in the graph itself. Selecting an object opens a Contextual Toolbar just above the graph, which gives information about the properties of that object. In this instance we see that plot1 is a range-area or rarea plot, with color Gray 13, intensity 80% and outline width Medium. If we click More... on the Contextual Toolbar we would see further options to control the properties of the selected object.

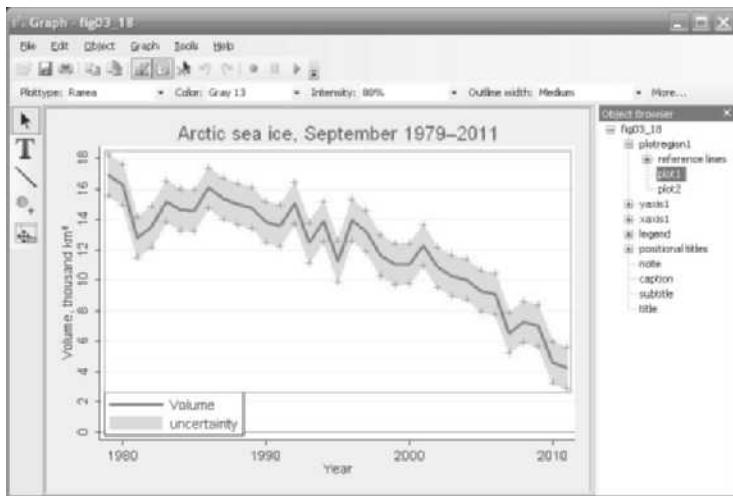


Figure 3.29

Changing the plottype from Rarea (range area) to Rcap (range capped), color from Gray 13 to Gray 3 (darker), and outline width from Medium to Medium thick gives the graph a new look (Figure 3.30).

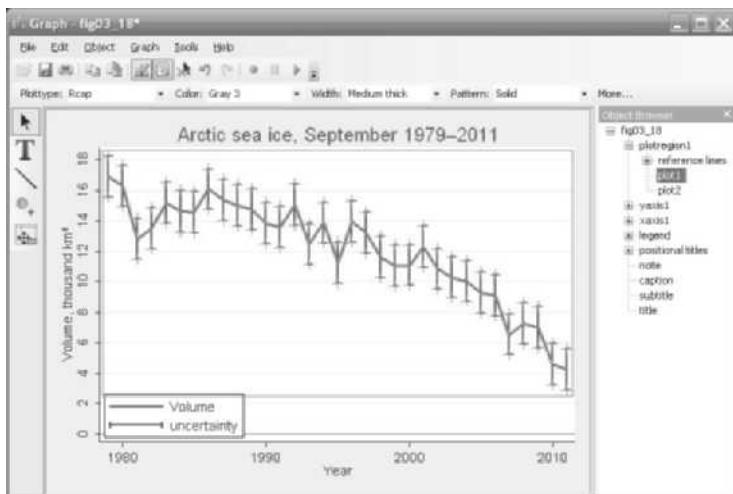


Figure 3.30

Plot2 in this image refers to the **twoway line** plot that was overlaid to form the main line in Figure 3.18. If we select plot2 in the Graph Editor, then change its plottype from Line to Scatter, and its color to Gray 5, we get something like Figure 3.29. This final image also includes an arrow with the text “2010 significantly lower,” added using the T (text) and line tools from the left-margin toolbar.

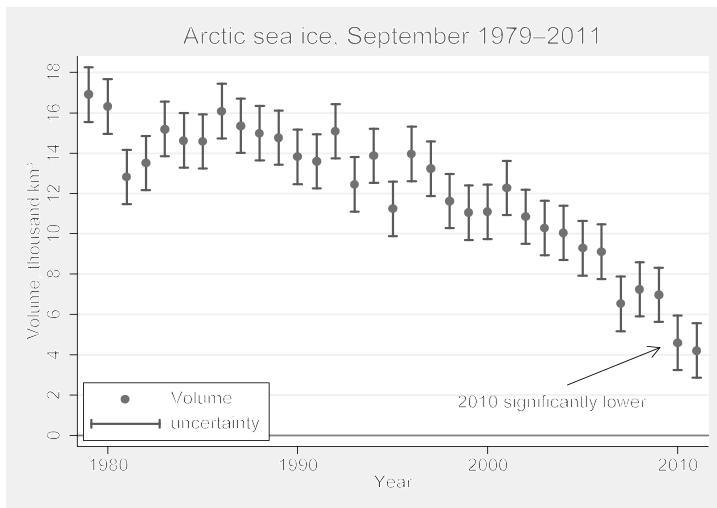


Figure 3.31

In general, the Graph Editor revises graph features in ways that could have been controlled by the original **graph** command. We cannot do things such as move an individual data point in the plot, although we can add or move new markers at any position. On the other hand, it is easy to change the properties of markers, lines, axis labels or titles. We can also hide objects, making them invisible. Any changes made in the Graph Editor become permanent when we save the graph.

Creative Graphing

Edward Tufte, in his elegant and influential books about graphing data (1990, 1997, 2001, 2006), calls for more effort at designing clear, information-packed graphics. Presenting a rich collection of impressively good or humorously awful examples, Tufte shows how successful graphics allow viewers to draw their own comparisons and examine details of relationships between variables. Stata users form a natural audience for these suggestions. Stata provides flexible tools for visualizing patterns in complex data, allowing basic plots to be enhanced or rearranged creatively in new images.

One of Tufte's themes is the value of small multiples, sets of thumbnail-sized graphics that add dimensions for comparison. A **graph** command with **by()** option can draw these nicely. Figure 3.32 illustrates with time plots of winter snow depth at two locations: a town in New Hampshire's White Mountains, and the city of Boston, 225 kilometers to its south (dataset *whitemt1.dta*). Snow depth was measured daily at both locations; these data cover nine consecutive winters, 1997–98 through 2005–2006. Variable *dayseason* counts days since November 1 each winter season. *mtdepth* and *bosdepth* are snow depths in centimeters at the White Mountains location and in Boston, respectively. Variable *season* identifies the winter seasons, 1997–98 through 2005–06. The following command specifies a **twoway area** graph of *mtdepth* and *bosdepth* against *dayseason*, using lighter and darker gray (gs11 and gs5) for

colors, in a 3×3 layout by *season*, and with a one-column legend positioned at 3 o'clock. `symxsize(*.3)` saves space by making the legend's symbols only 30% as wide as their default.

```
. graph twoway area mtdepth bosdepth dayseason
    if dayseason>29 & dayseason<160,
    bcolor(gs11 gs5) ytitle("Snow depth, cm")
    by(season, rows(3) note("") legend(position(3)))
    xlabel(30(30)150) ylabel(0(20)80)
    legend(cols(1) label(1 "White Mt") label(2 "Boston"))
    symxsize(*.3))
```



Figure 3.32

Figure 3.32 visualizes daily conditions through nine New England winters, showing how snow depth varies at two different places and on two different time scales. 2000–01 and 2003–04 stand out as heavy snow seasons in the mountains, with several significant storms in Boston. 1998–99 was much lighter in the mountains, with periods of no snow on the ground.

The data behind Figure 3.32 were assembled for research on how weather and climate affect attendance at ski areas (Hamilton et al. 2003, 2007). As New England's winter climate warmed in recent decades, low-snow winters became more common. That warming is troublesome from environmental and other perspectives, including that of winter recreation. Ski areas can feel not only the effects of local snow conditions, but also a “backyard effect” of snow conditions in distant cities such as Boston, where many skiers live. The next graph, Figure 3.33, focuses on the single season of 1999–2000 (dataset *whitemt2.dta*). It begins with the same snow-depth shadow mountains that formed the top right plot in Figure 3.32.

Figure 3.33 overlays these shadow mountains (the **twoway area** plot) with a **line** plot showing the number of skier and snowboarder visits each day, at one ski area in the White Mountains close to where the snow-depth measurements were made. Both the observed visits (*visits*) and the number of visits predicted by a time series model (*model*) are graphed. The model, described

in Hamilton et al. (2007), predicts daily attendance as a function of weekly cyclical factors together with weather and snow conditions, both in the mountains and in Boston. The **graph** command creating Figure 3.33 assigns the left-hand *y* axis to snow depth in centimeters (*mtdepth* and *bosdepth*), and the right-hand *y* axis to observed and modeled number of visitors (*visits* and *model*).

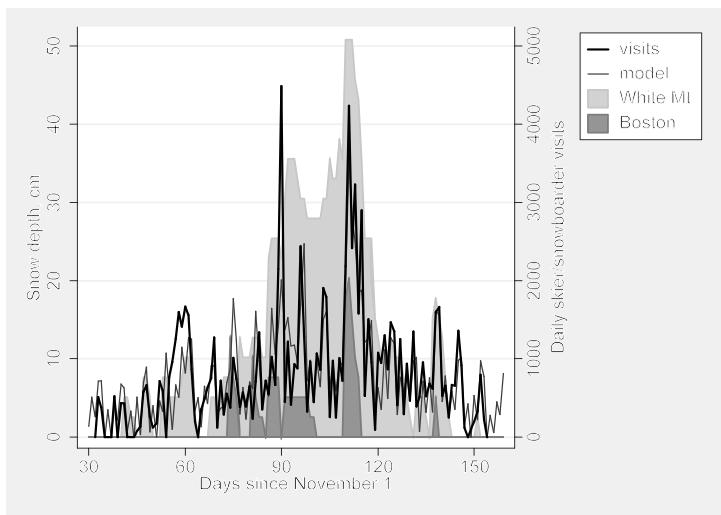


Figure 3.33

Note that by carefully setting the **yscale(range())** and **ylabel()** options for each of the two overlaid plots in Figure 3.33, we managed to align their scales so that the same horizontal grid lines work for both. This is not practical to do with all data, but can definitely improve the readability of graphs involving differently-scaled *y* variables.

```
. graph twoway area mtdepth bosdepth dayseason, yaxis(1)
    ytitle("Snow depth, cm", axis(1)) bcolor(gs12 gs6)
    ylabel(0(10)60, axis(1))
    || line model visits dayseason, yaxis(2)
    lwidth(medthin medthick)
    ylabel(0(1)3, axis(2)) lcolor(gs1 gs0)
    || if dayseason>29 & dayseason<160,
    r2("Daily skier/snowboarder visits") xlabel(30(30)150)
    xtitle("Days since November 1")
    legend(rows(4) position(2) order(4 3 1 2) label(1 "White Mt")
        label(2 "Boston") label(3 "model") label(4 "attend")
        symxsize(*.3))
    yscale(range(0,51) axis(1)) ylabel(0(10)50, axis(1) grid)
    yscale(range(0,5100) axis(2)) ylabel(0(1000)5000, axis(2))
```

The two highest spikes in ski-area visits were school holiday periods that happened to coincide with snow in Boston. The original study tested and confirmed the significance of this backyard effect. Graphically, it would be a simple step (not shown) from Figure 3.33 to a new set of small-multiples plots like Figure 3.32, visualizing the ski business together with the snow.

Population pyramids, widely used by demographers to represent the age-sex structure of populations, are not among Stata's plot types. They can, however, be constructed from horizontal bar charts, through creative use of **graph hbar**. There are several ways to do so. Figure 3.34 illustrates one approach, with a pyramid for the Greenland-born, predominantly Inuit, population of Greenland in 2006 (Hamilton and Rasmussen 2010). The number of females at each age is indicated by a bar to the right of center, and the number of males that same age by a bar to the left. The 90 one-year age groups seen here are too many to label individually, so they are marked off instead by gray bands every 20 years (0–19 years, 20–39 years, etc.). The graph indicates, for example, that in 2006 the Greenland-born population included almost 600 40-year-old males but fewer than 500 40-year-old females, reflecting sex differences in net outmigration. The central bulge in this pyramid marks a baby boom of adults now ages 35–49 (born in the 1950s and 60s), followed by much smaller cohorts of younger adults. We also see an echo boom of children, born in the 1980s and 90s to adults from the first baby boom. Ages 10–14 comprise the most numerous cohort among children.

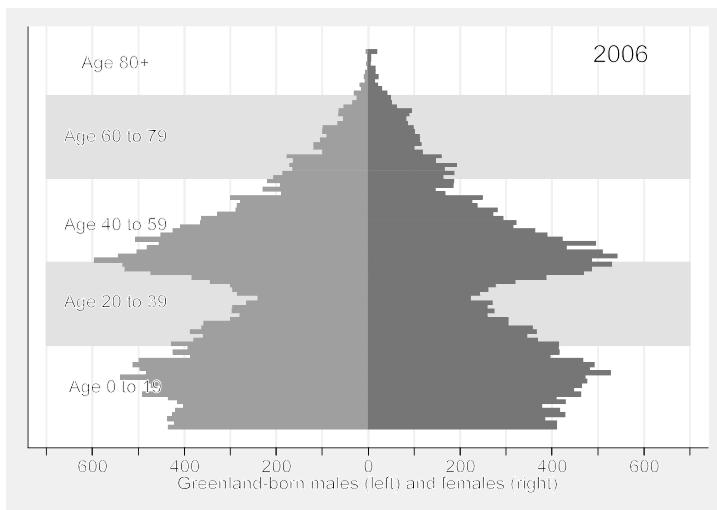


Figure 3.34

There are several tricks behind Figure 3.34. The raw data (*greenpop1.dta*) contain counts of the number of *males* and *females* at each *age*. In order to graph males on the left, we generate a new variable equal to the negative of the number of males,

```
. gen negmales = -males
```

A basic unlabeled pyramid could then be drawn by a command such as

```
. graph hbar (sum) negmales females if year==2006,  
over(age, descending gap(0) label(nolabel))
```

To place gray bands in the background, marking off 20-year groups, we define fake variables *maleGRAY* and *femGRAY* just to fill in the graph to plus or minus 700:

```
. gen maleGRAY = -(700-males) if (age>=20 & age<40)
| (age>=60 & age<80)
. gen femGRAY = 700-females if (age>=20 & age<40)
| (age>=60 & age<80)
```

Figure 3.34 now can be drawn by stacking *negmales*, *females*, *maleGRAY* and *femGRAY* in a horizontal bar chart, with text to label the gray bands. We also apply labels such as “600” for –600 on the y axis so that the counts for males do not appear negative.

```
. graph hbar (sum) negmales females maleGRAY femGRAY if year==2006,
over(age, descending gap(0) label(nolabel))
ylabel(-600 "600" -400 "400" -200 "200" 0 200 400 600)
ytick(-700(100)700, grid) legend(off) stack
ytitle("Greenland-born males (left) and females (right)")
bar(1, color(emidblue)) bar(2, color(maroon)) bar(3, color(gs14))
bar(4, color(gs14)) text(550 97 "2006", size(large))
text(-550 11 "Age 0 to 19")
text(-550 33 "Age 20 to 39") text(-550 53 "Age 40 to 59")
text(-550 76 "Age 60 to 79") text(-550 95 "Age 80+")
```

Figure 3.35 takes this idea a step further by showing similar age pyramids for 1977, 1986, 1996 and 2006. In this sequence we can follow the rise of the large cohort born following improvements in Greenlanders’ health and living standards in the 1950s and 60s. This baby boom shows up as teenagers in the 1977 pyramid. As the boom generation enters adulthood by the 1986 pyramid, we see the echo boom of their children. By 2006, this echo boom is waning.

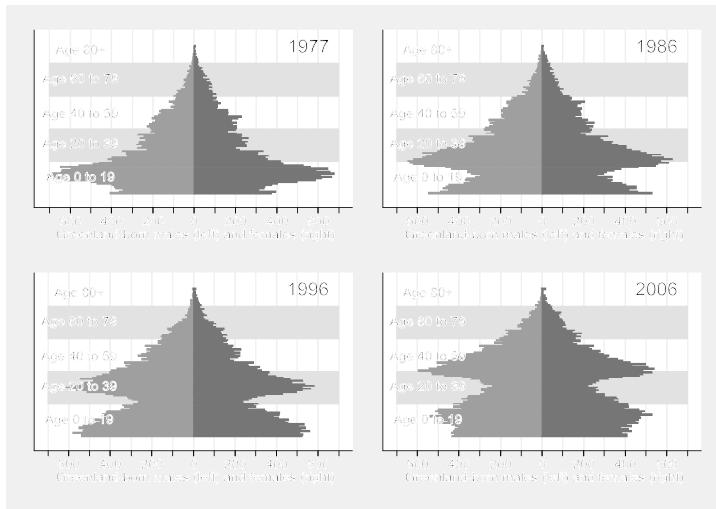


Figure 3.35

Although Figure 3.35 (constructed using separate images and **graph combine**) follows a small-multiples idea similar to Figure 3.32, these pyramids can be displayed in a more interesting way. For a live presentations I drew a set of 30 annual pyramids, 1977 to 2006, using a do-file. These 30 Stata graphs (in .emf format) were then pasted onto one PowerPoint slide each, with

automatic transitions at 1-second intervals, producing a 30-second animation of Greenland's demographic change. Another animation showed how the population of non-Greenlanders living in Greenland had changed over the same years, an interconnected but quite different demographic tale (Hamilton and Rasmussen 2010).

Figure 3.36 is less dynamic, but combines five simple plots with text to form an image having some properties of both an illustration and a table. The resulting Stata graphic depicts population changes 1990–2000 among different ethnic groups living in rural counties of the U.S. South (based on U.S. Census data assembled by Voss et al. 2005). The left side of Figure 3.36 is a **twoway area** graph. To achieve the ramped effect showing population change, the variables graphed for each ethnic group (*popwbho*, *popwbh* etc.) actually represent sums calculated as that group's population plus all the other populations that are graphically "below" it (dataset *southmig1.dta*). Important additional information, not evident from the area plot itself, is conveyed by two lines of labeling for each group in the legend. For example, readers can see from the legend that the Hispanic population of the rural South grew by 61% over this decade, from roughly 800,000 to 1.3 million people, and make their own visual or numerical comparisons with other populations.

```
. graph twoway area popwbho popwbh popwb popw year,
    legend(rows(4) position(3) symxsize(3)
    label(1 "Other, +50%" "0.3 to 0.5m")
    label(2 "Hispanic, +61%" "0.8 to 1.3m")
    label(3 "Black, +7%" "3.6 to 3.8m")
    label(4 "White, +6%" "14.9 to 15.7m"))
    xlabel(1990 2000) xtitle("")
    ylabel(0(5)20,angle(horizontal) grid)
    ytitle("Population in millions")
    title("Population growth 1990-2000")
```

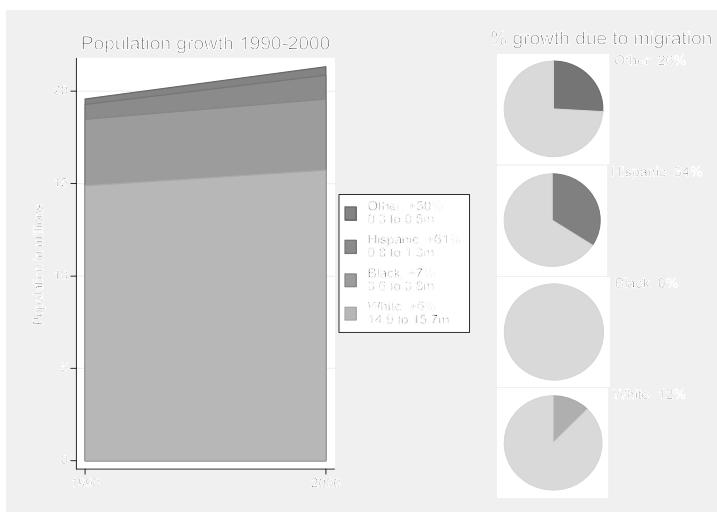


Figure 3.36

The right-hand part of Figure 3.36 consists of four pie charts showing the percentage of population growth due to net in-migration. Each pie chart was drawn separately using dataset *southmig2.dta*. For example, the bottom pie chart shows that 12% of the white population growth reflects net migration. Variables graphed are net migration (*netmig_w*, the total number of in-migrants minus out-migrants) and the remainder of population growth due to natural increase (*nonmig_w*, number of births minus deaths).

```
. graph pie nonmig_w netmig_w,
    legend(off) pie(1, color(dkorange)) pie(2, color(gs13))
    title("White 12% ", position(2))
```

Each individual pie chart was saved with a file name such as *pie_white.gph*. After drawing and saving four such pie charts, they were brought together using **graph combine**.

```
. graph combine pie_other.gph pie_hisp.gph
    pie_black.gph pie_white.gph,
    imargin(tiny) rows(4)
    title("% growth due to migration") fysize(40)
```

An **fysize(40)** option forces this four-pie-chart image to use only 40% of the width available. Consequently, when they are combined with the left-hand area plot to make Figure 3.36, the pie charts take up less than half of the total width.

These examples illustrate some of the potential for designing new graphics in Stata, by combining standard elements in fresh ways.

Survey Data

This chapter provides a brief introduction to working with survey data in Stata. Readers not interested in surveys could skip this chapter with no loss of continuity. On the other hand, those in social sciences where survey data have central importance may benefit from a quick look at how Stata handles such data. More advanced analytical methods will come up in later chapters.

Survey research places great emphasis on drawing valid inferences about populations. Simple random sampling is theoretically best for this purpose, but often that is impossible or too expensive. In place of the simple random-sampling ideal, more convenient but sometimes complex sampling strategies are employed to achieve reasonably representative samples, and post-sampling adjustments applied where needed. Because standard statistical methods assume simple random sampling, we also need specialized methods designed for survey data that can take information about the sampling process into account.

Stata has developed a reputation for strength in survey data analysis, based on a unified approach encompassing a wide selection of analytical methods. All of these methods work from an underlying definition of the sample structure, which can include probability weights adjusting for design or selection bias. The sampling design could also involve complexities such as stratification, multi-stage clustering, finite populations at one or more levels, with-replacement sampling, replication weights, or poststratification. Once the basic design elements have been specified (via the `svyset` command), a dataset is saved with this information. Subsequent analyses using `svy:` commands will automatically apply weights and other survey design information as appropriate.

Most Stata survey procedures can be called using menus, and especially the many submenus of Statistics > Survey data analysis . Typing `help survey` brings up a starting place with links to further information. The *Survey Data Reference Manual* provides examples and technical details for the full set of Stata survey-related commands. Other good references include a book on sampling by Levy and Lemeshow (1999), and on the analysis of survey data, with biostatistical examples, by Korn and Graubard (1999). Sul and Forthofer (2006) provide a brief overview of some main issues in survey data analysis. See Moore (2008) for a well-illustrated discussion of some pitfalls in survey question wording.

Example Commands

- **svyset _n [pweight = *censuswt*]**
Declares dataset as survey type, with probability weights (proportional to the inverse of the probability of selection) given by variable *censuswt*. The *_n* specifies individual observations (the default) as primary sampling units or PSUs.
- **svyset _n [pweight = *censuswt*], strata(*district*)**
Declares dataset as survey type, from one-stage stratified sampling: the population was partitioned into strata, and individuals sampled independently within each strata. In this example, variable *district* identifies the strata, and *censuswt* the probability weights.
- **svyset school [pweight = *finalwt*], fpc(*nschools*)
|| _n, fpc(*nstudents*)**
Declares dataset as survey type from two-stage cluster sampling. In the first stage, schools were selected randomly, so schools are the PSUs. In the second stage, students were selected at random within the selected schools. Finite population corrections (FPCs) are specified for each stage: *nschools* is the total number of schools in the population, and *nstudents* is the number of students in each school.
- **svy: tabulate *vote*, percent miss ci**
Obtains a table of weighted percentages and confidence intervals for variable *vote*, including the missing values, based on **svyset** data.
- **svy: tabulate *vote* *gender*, column pearson lr wald**
Obtains a weighted cross-tabulation of *vote* by *gender*, with proportions based on each column (*gender*). Pearson χ^2 , likelihood-ratio χ^2 and Wald test statistics are reported.
- **svy: regress *y* *x1* *x2* *x3***
Performs survey regression of *y* on three predictors, *x1*, *x2* and *x3*. Type **help svy estimation** for a complete listing of regression-type survey estimation procedures.
- **svy, subpop(*voted*): regress *y* *x1* *x2* *x3***
Performs survey regression analysis using only a subpopulation defined by 1 values of the {0,1} variable *voted*. Selecting subsets of the data in the usual way with **if** or **in** statements is not appropriate for most survey analysis; use a **svy, subpop()**: option instead.
- **svy: mean *age*, over(*gender*)**
Finds survey weighted means and confidence intervals for *age*, over categories of *gender*.

Declare Survey Data

Since 2001, the Granite State Poll at the University of New Hampshire has conducted statewide telephone surveys several times each year. Each survey contacts a new sample of about 500 people, asking a variety of opinion questions along with respondent background characteristics. The poll's political findings attain national importance every four years during New

Hampshire's presidential primary campaigns. Dataset *Granite_2011_6.dta* contains questions from a Granite State Poll of 516 people, conducted in June 2011.

```
. use C:\data\Granite2011_6.dta, clear
. describe, short

Contains data from C:\data\Granite2011_6.dta
obs:      516
          New Hampshire, Granite State
vars:      33
          Poll -- June 2011
size:    19,608
          18 Mar 2012 15:46
Sorted by:  respnum
```

As with any survey, the sampling design and response patterns may result in sample data that differs from the target population. For example, Census data tell us that less than 52% of the state's adult population is female, but women make up almost 55% of this sample.

```
. tab sex
```

Gender	Freq.	Percent	Cum.
Male	234	45.35	45.35
Female	282	54.65	100.00
Total	516	100.00	

To compensate for minor bias in sampling, survey researchers routinely calculate probability weights. Some weights are calculated from comparisons between sample and population characteristics, such as gender in the example above. Others are based on features of the sampling design. For the Granite State Poll, researchers define a variable named *censuswt* that combines adjustments for household size, number of telephone lines, gender and region within the state. *censuswt* values in the June 2011 poll have a mean of 1 but range from .16 (observations given low weights to compensate for over-representation) to 2.19 (high weights to compensate for under-representation).

```
. summarize censuswt
```

Variable	Obs	Mean	Std. Dev.	Min	Max
censuswt	516	.9991743	.4601123	.1603937	2.194549

A *svyset* command declares the survey structure of the data, with probability weights given by *censuswt*. Saving the data then saves this information, although survey weights will be used in any particular analysis only if we specifically ask for them. Otherwise the data are unchanged.

```
. svyset _n [pweight = censuswt]

pweight: censuswt
VCE: linearized
Single unit: missing
Strata 1: <none>
SU 1: <observations>
FPC 1: <zero>

. save, replace
. svydescribe
```

Survey: Describing stage 1 sampling units

```
pweight: censuswt
VCE: linearized
Single unit: missing
Strata 1: <none>
SU 1: <observations>
FPC 1: <zero>
```

Stratum	#Units	#Obs	#Obs per Unit		
			min	mean	max
1	516	516	1	1.0	1
1	516	516	1	1.0	1

Once data have been **svyset**, commands with the prefix **svy:** will perform calculations using the survey weight information. After weighting the gender balance is closer to what we expect in the population.

```
. svy: tab sex
(running tabulate on estimation sample)

Number of strata      =           1
Number of PSUs         =         516
                                         Number of obs      =       516
                                         Population size   = 515.57392
                                         Design df        =       515


```

Gender	proportions
Male	.4965
Female	.5035
Total	1

Key: proportions = cell proportions

Many Stata commands from simple tables to statistical models permit the **svy:** prefix. For example, we could perform a weighted logistic regression (Chapter 9) of personal opinions about climate change, variable *warmop2*, on respondent education and political party through a command such as

```
. svy: logit warmop2 educ party
```

Type **help survey** for a list of analytical possibilities. The **svyset** command also can declare more information than just the probability weights used in the example above. **svyset** options allow for complex designs including stratified and multistage cluster sampling, finite population corrections, alternative methods of variance estimation, and poststratification. Type **help svyset** to see the syntax and a complete list of options. The *Survey Data Reference Manual* gives more examples and technical details.

Design Weights

The previous section took weight definitions for granted, and indeed many data users begin with a completed survey in which weights have already been calculated by someone else. This and the following sections present examples showing how such calculations are done.

Survey researchers apply probability weights to adjust for biases in their sampling methods. The biases could arise from intentional features of the sampling design, or from accidental properties of the data-collection process. Either way, initial sampling yields data not well representing the population of interest. Probability weighting attempts to compensate for departures from simple random sampling, and give us a more realistic picture of sampling variability and population characteristics.

For the Granite State Poll, interviewers call a random sample of New Hampshire household telephone numbers. In theory, these random phone numbers could yield a representative sample of households. For voting research and many other purposes, however, pollsters want to generalize not about the population of households, but about the population of adults (or voters) living in those households. Some households contain only one adult, whereas others have more. Among respondents to the June 2011 poll, about 29% said that they lived in a one-adult household. Responses in this example are limited to one, two, or three or more adults, as a practical compromise for weighting purposes. Only 503 of the original 516 respondents gave an answer regarding the number of adults; we will return to the other 13 later.

. tab adults

# adults in household	Freq.	Percent	Cum.
1	148	29.42	29.42
2	273	54.27	83.70
3+	82	16.30	100.00
Total	503	100.00	

Although 29% of our sample lived in one-adult households, it would be a mistake to guess that a similar percentage of New Hampshire adults do so. To select a resident randomly, once a household has been called, the telephone interviewers ask to speak to the adult in the household who had the most recent birthday, or would call back later if that individual was not present. People from households with one adult therefore were at least three times more likely to enter our sample, compared with those from households with three or more. The table above suggest that our phone calls reached households with at least $(1 \times 148) + (2 \times 273) + (3 \times 82) = 940$ adults. Among this sample of pseudo-people, those living in one-adult households make up only $148/940$ or 16%, much less than the 29% in our table.

Survey weights provide a way to adjust for such known sampling biases, and achieve more realistic results. In this example that could matter not only for describing household size, but for other things such as voting behavior that might be correlated with household size. Single-adult households probably include a larger proportion of elderly people living alone. Two-adult households will include many young families. Multi-adult households will often be older families with adult children, or else young adults with roommates.

Probability weights are proportional to the inverse of the probability of selection. For our example, the conditional probability of selecting a particular person from a one-adult household (given that we phoned there) equals one. The probability of selecting a person from a two-person household equals $1/2$, and from a three-person household $1/3$. If we used the inverse of these probabilities, 1, 2 and 3, as weights then our sample would contain 940 pseudo-people—giving us the correct proportions, but leading to incorrect totals and other confusion. To

maintain the true sample size, we can multiply these inverse probabilities by the ratio of real people to pseudo-people, 503/940. This step creates a new variable named *adultwt*, which contains the probability weights to adjust for a known sampling bias, while preserving the original sample size. The weights are .535 (one-adult household), 1.070 (two adults) or 1.605 (three or more); missing values get a neutral weight of 1. The ratio of these weights remains 1:2:3.

```
. generate adultwt = adults*(503/940)
. replace adultwt = 1 if missing(adultwt)
. tab adults, summ(adultwt) miss
```

# adults in household	Summary of adultwt		
	Mean	Std. Dev.	Freq.
1	.53510636	0	148
2	1.0702127	0	273
3+	1.6053191	0	82
DK/NA	1	0	13
Total	.99999997	.35080553	516

If this were the only adjustment desired, we could **svyset** the data using *adultwt* as probability weights.

```
. svyset _n [pw = adultwt]
    pweight: adultwt
    VCE: linearized
    Single unit: missing
    Strata 1: <none>
    SU 1: <observations>
    FPC 1: <zero>

. svy: tab adults, percent
(running tabulate on estimation sample)

Number of strata      =           1
Number of PSUs        =         503
                                         Number of obs      =      503
                                         Population size  =  502.99998
                                         Design df        =      502

# adults in household |   percentages
-----|-----
1          |       15.74
2          |       58.09
3+         |       26.17
Total      |       100
Key: percentages = cell percentages
```

The weighted proportions (such as 16% from one-person households, instead of 29% as in the raw data) give a more realistic picture.

Poststratification Weights

The previous section gave an example of weights based on the sampling design, which was known before data collection began. A second type of weights might be defined after we have collected the data, and see that despite our best efforts, it appears unrepresentative in some respect. For instance, the sample might have a gender or age distribution noticeably different from those of our target population, making other results suspect. Poststratification refers to probability weights calculated so that the proportions of particular groups or strata in our sample more closely resemble the population.

The Granite State Poll sample is 54.65% female. But according to the 2010 Census, the adult population of New Hampshire is only 51.6% female. If we guessed from the survey that the state's population was something close to 54.65% female, we would be off the mark. Moreover, we could easily draw mistaken conclusions about other things correlated with gender, such as voting. This apparent response bias could undermine our ability to draw inferences about a larger population.

. tab sex

Gender	Freq.	Percent	Cum.
Male	234	45.35	45.35
Female	282	54.65	100.00
Total	516	100.00	

There are many ways to approach poststratification. (For an alternative to the by-hand approach shown below, Stata's `svyset` command offers a `poststrata` option, illustrated in the *Survey Reference Manual*; type `help svyset` for the basic syntax.) If we know the true population percentages of key variables, as we do regarding gender, then weights to adjust for response bias can be calculated from population percentage divided by sample percentage. `Sex` is coded 0 for males, who make up 48.4% of the adult population of New Hampshire but only 45.35% of this sample. It is coded 1 for females, who are 51.6% of the population and 54.65% of the sample. There are no missing values of `sex` in these data.

We calculate weights slightly above one for males: $48.4/45.35 = 1.067$. For females the weights are slightly below one: $51.6/54.65 = .944$.

```
. generate sexwt = 48.4/45.35 if sex==0
. replace sexwt = 51.6/54.65 if sex==1
. tab sex, summ(sexwt)
```

Gender	Summary of sexwt		Freq.
	Mean	Std. Dev.	
Male	1.0672547	0	234
Female	.94419032	0	282
Total	.99999857	.06132481	516

If we `svyset` the data using `sexwt` as the probability weight, `svy: tab` produces a weighted table showing exactly the right proportions of men (48.4%) and women (51.6%). After calculating poststratification weights, it is good to check on whether they perform as intended.

```
. svyset [pw = sexwt]
  pweight: sexwt
  VCE: linearized
  Single unit: missing
  Strata 1: <none>
  SU 1: <observations>
  FPC 1: <zero>

. svy: tab sex, percent
(running tabulate on estimation sample)

Number of strata      =           1
Number of PSUs        =         516
                                         Number of obs      =       516
                                         Population size   = 515.99926
                                         Design df        =       515



| Gender | percentages |
|--------|-------------|
| Male   | 48.4        |
| Female | 51.6        |
| Total  | 100         |


Key: percentages = cell percentages
```

More elaborate poststratification weights could follow a similar approach. For example, suppose that for a different study we wish to approximate a population age-race-sex distribution.

1. Obtain a table of age-race-sex percentages from Census or other data on the population of interest, such as adults living in a particular state. If we employ five groups for age (18–29, 30–39, etc.) and two for race (white, nonwhite), this results in 20 numbers such as the percentage of that state's adult population consisting of white males 18–29, the percentage of white females 18–29, and so forth.
2. Obtain a similar table of age-race-sex percentages from the sample, for example by creating and tabulating a new variable named *ARS* denoting age-race-sex combinations:

```
. egen ARS = group(agegroup race sex), lname(ars)
. tab ARS
```

3. Define a new set of weights, using **generate ... if** commands. For example, suppose we know that 8.6% of the adult population in the study area consists of white males age 18–29, and 8.2% are white females in this age group. In our unweighted sample, however, we see only 2.6% white males 18–29, and 5.1% white females — so the young adults, particularly young males, are under-represented. We could create a new age-race-sex weight variable named *ARSwt* equal to 1 (a neutral weight) if we do not know a respondent's age-race-sex combination, and otherwise equal to the population percentage divided by corresponding sample percentage for their age-race-sex group. The first few commands could be

```
. generate ARSwt = 1 if ARS >= .
. label variable ARSwt "Age-race-sex weights"
. replace ARSwt = 8.6/2.6 if ARS == 1
. replace ARSwt = 8.2/5.1 if ARS == 2
```

Poststratification adjustments work best in connection with carefully-designed surveys, and should not be misunderstood as a cure for haphazard sampling. Such adjustments have been applied most extensively in areas such as voter opinion polls and social science surveys, where great effort goes into securing the most representative samples to begin with. These also are areas where independent evidence such as vote outcomes or replications by other researchers provides reality tests for how well the adjustments succeed.

A single dataset might include weight variables calculated from more than one source, such as design weights and poststratification weights. To combine these into one overall weight variable, we multiply and then make an adjustment so that the final sum of weights equals the sample size. A **quietly** prefix before **summarize** tells Stata to calculate summary statistics but do not show us the results, to save time or space. The **quietly** prefix works similarly with other commands as well.

```
. generate finalwt = adultwt*ARSwt
. replace finalwt = 1 if finalwt >= .
. quietly summarize finalwt
. replace finalwt = finalwt*(r(N)/r(sum))
```

Any number of weight variables can exist in the same dataset, and **svyset** used repeatedly to select among them as needed for particular analysis. The weights affect analysis only when we apply them through **svy:** or other explicitly weighted commands. For the remainder of this chapter, we return to the Granite State Poll weighted by *censuswt*, a variable calculated by the UNH Survey Center to combine design weighting (for number of adults, and number of phone lines) with poststratification (for gender and region within New Hampshire).

```
. svyset _n [pw = censuswt]
pweight: censuswt
VCE: linearized
Single unit: missing
Strata 1: <none>
SU 1: <observations>
FPC 1: <zero>
```

Survey-Weighted Tables and Graphs

The June 2011 Granite State Poll included six questions related to global warming or climate change. Several questions were factual, but one (*warmop*) asked what you personally believe.

Which of the following three statements do you personally believe?

Climate change is happening now, caused mainly by human activities.

Climate change is happening now, but caused mainly by natural forces.

Climate change is NOT happening now.

Interviewers rotated the order of response choices to avoid possible bias. About 55% agree that climate change is happening, now, caused mainly by human activities. Others think the changes have mainly natural causes (35%). Few believe that climate change is not happening now (3%).

```
. svy: tab warmop, percent ci
(running tabulate on estimation sample)

Number of strata      =           1
Number of PSUs        =         516
                                         Number of obs      =       516
                                         Population size = 515.57392
                                         Design df        =       515

Personal belief about climate change |   percentages          lb          ub
DK/NA                         7.443      5.252      10.45
Not now                      3.05       1.904      4.853
Now/natu                     34.62      30.2       39.32
Now/huma                     54.89      50.11      59.58
Total                         100

Key: percentages = cell percentages
     lb        = lower 95% confidence bounds for cell percentages
     ub        = upper 95% confidence bounds for cell percentages
```

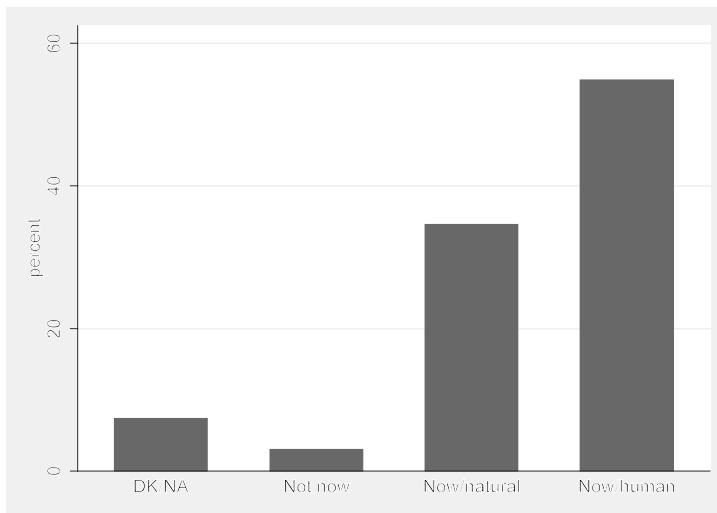
The **svy: tab** command applies weights according to the specification declared earlier by **svyset**. **ci** requests confidence intervals for the weighted percentages, shown as lower and upper bounds (lb and ub). Based on this sample, we are 95% confident that between 50.11% and 59.58% of New Hampshire adults believe that human activities are changing the climate.

Stata's native graph types are not ideal for viewing categorical-variable distributions such as those in the table above. Fortunately, a user-written program named **catplot**, introduced in a *Stata Journal* article (Cox 2004b), does this job quite well. You can obtain the ado-files for **catplot** easily from the Web by typing

```
. findit catplot
```

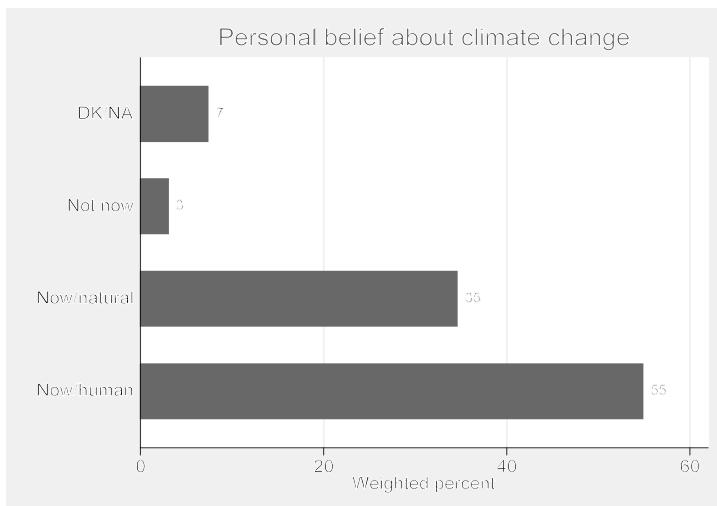
and following the links to install them on your computer. (The **findit** command works for hundreds of other user-written programs as well.) Once it is installed, typing **help catplot** will show the command's syntax and options. Figure 4.1 contains a simple **catplot** bar chart of *warmop*. Although **catplot** does not accept pweights, specifying [**aweights = censuswt**] in this command will have the same visual effect so that bar heights correspond to the **svy: tab** percentages.

```
. catplot bar warmop [aweight = censuswt], percent
```

**Figure 4.1**

Bar charts with value labels often are easier to read in a horizontal-bar (**hbar**) format, particularly when we have many bars. Figure 4.2 shows a horizontal version including a title and better axis label, suitable for a report or presentation on the survey results. We also label the bars so that weighted percentages can be read directly from the graph; these are the same numbers found by **svy: tab** above.

```
. catplot hbar warmop [aweight = censuswt], percent
    blabel(bar, format(%3.0f)) ytitle("Weighted percent")
    title("Personal belief about climate change")
```

**Figure 4.2**

How do climate change beliefs relate to other variables in the survey, such as respondent education (*educ*)? We could answer such questions through a two-way tabulation,

```
. svy: tab warmop educ, col percent
(running tabulate on estimation sample)

Number of strata      =           1
Number of PSUs        =         511
                                         Number of obs      =      511
                                         Population size   =  510.02315
                                         Design df        =      510
```

Personal belief about climate change	Highest degree completed				Total
	HS or le	Tech/som	College	Postgrad	
DK/NA	9.946	12.27	4.603	4.041	7.524
Not now	5.154	2.694	2.694	1.991	3.083
Now/natu	33.55	42.81	37.29	24.79	34.71
Now/huma	51.35	42.23	55.41	69.18	54.68
Total	100	100	100	100	100

Key: column percentages

Pearson:
Uncorrected $\chi^2(9) = 25.1986$
Design-based $F(8.81, 4495.16) = 2.4226 \quad P = 0.0102$

In the example above we asked for column percentages, because the column variable (*educ*) forms the independent variable for this analysis. About 69% of those with postgraduate degrees, 55% of college graduates, and 42% with some college or technical school personally believe that climate change is happening now, due mainly to human activities. The design-weighted *F* test for this table confirms that the relationship between climate belief and education is statistically significant ($p = .0102$).

Figure 4.3 shows a **catplot** of *warmop* responses by education, giving the same weighted percentages. The **percent(educ)** option calls for percentages within categories of *educ*. We also place the **title()** as a sub-option within **by()** in order to get a single title for the whole graph. You can experiment to see what happens without these options.

```
. catplot hbar warmop [aweight = censuswt], percent(educ)
    by(educ, title("Personal belief about climate change"))
    blabel(bar, format(%3.0f)) ytitle("Weighted percent")
```

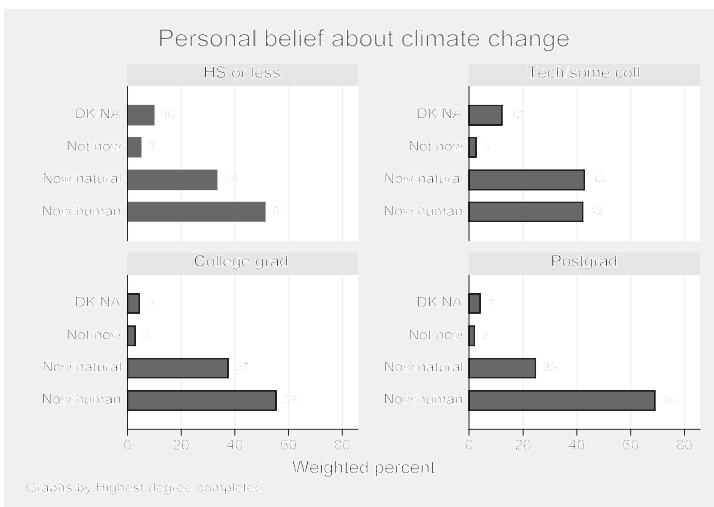


Figure 4.3

Bar Charts for Multiple Comparisons

The **catplot** bar charts in Figure 4.3 depict a relationship between two categorical variables, each with four categories. If we have more than two variables, or more than a few categories, however, the **catplot** approach becomes cluttered. A cleaner alternative for making multiple comparisons of categorical variables employs Stata's horizontal bar chart command **hbar**.

Figure 3.13 in the previous chapter tracked changes in the late-summer area of Arctic sea ice, over the years 1979–2011. The dramatic decline has attracted much scientific attention, and been widely noted in popular news media as well. Our Granite State Poll included a carefully worded question (*warmice*) testing whether people know about this. As with *warmop*, the order of response choices was rotated to avoid bias. A large majority (71%) know that ice area declined.

Which of the following three statements do you think is more accurate?
Over the past few years, the ice on the Arctic Ocean in late summer ...
Covers less area than it did 30 years ago.
Declined but then recovered to about the same area it had 30 years ago.
Covers more area than it did 30 years ago.

```
. svy: tab warmice, percent ci
```

(running tabulate on estimation sample)

Number of strata	=	1	Number of obs	=	516
Number of PSUs	=	516	Population size	=	515.57392
			Design df	=	515

Arctic ice vs. 30 years ago			
	percentages	1b	ub
Less	70.91	66.35	75.08
Recovered	10.43	7.784	13.83
More	6.916	4.841	9.789
DK/NA	11.75	8.991	15.21
Total	100		

Key: percentages = cell percentages
 1b = lower 95% confidence bounds for cell percentages
 ub = upper 95% confidence bounds for cell percentages

The *warmice* question allows four response choices, including “don’t know” or no answer. For some purposes, however, it proves useful to create a new dichotomous variable that just indicates whether they answered this question correctly. Variable *warmiceQ* equals 1 for respondents who said that ice area declined, and 0 for everyone else. About 71% answered correctly that late-summer sea ice covers less area than it did 30 years ago.

```
. gen warmiceQ = 0
. replace warmiceQ = 1 if warmice==1
. label variable warmiceQ "Know Arctic ice area declined"
. svy: tab warmiceQ, percent ci
(running tabulate on estimation sample)
```

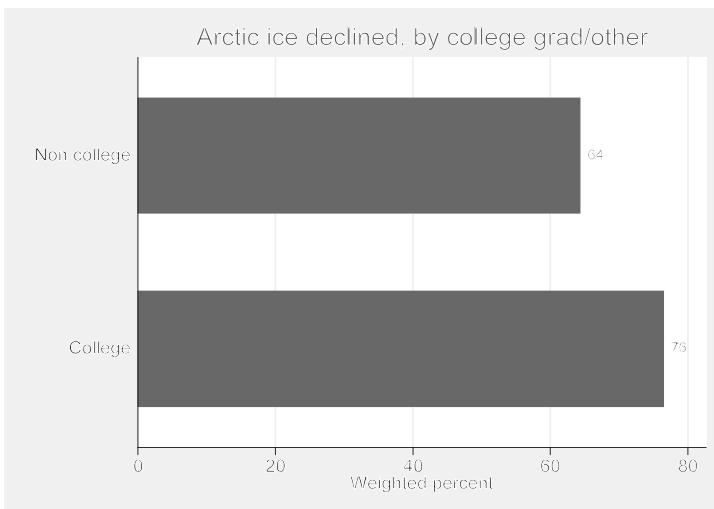
Number of strata	=	1	Number of obs	=	516
Number of PSUs	=	516	Population size	=	515.57392
			Design df	=	515

Know Arctic ice area declined			
	percentages	1b	ub
0	29.09	24.92	33.65
1	70.91	66.35	75.08
Total	100		

Key: percentages = cell percentages
 1b = lower 95% confidence bounds for cell percentages
 ub = upper 95% confidence bounds for cell percentages

The mean of a {0,1} variable like *warmiceQ* simply equals the proportion of 1's. Such {0,1} dummy variables have many uses in statistical modeling. For graphical purposes we might temporarily re-scale *warmiceQ* to have values of 0 or 100 instead. The mean of a {0,100} variable equals the percentage of 100s, or the percentage of correct answers on Arctic ice in this example. By applying **graph hbar** to a {0,100} dichotomy, we can visually compare many different percentages. The bar chart in Figure 4.4 gives the weighted percent of correct answers among college graduates and others (variable *college*).

```
. gen warmiceQ100 = warmiceQ*100
. graph hbar (mean) warmiceQ100 [aw = censuswt], over(college)
  blabel(bar, format(%3.0f)) ytitle("Weighted percent")
  title("Arctic ice declined, by college grad/other")
```

**Figure 4.4**

Note that in Stata's horizontal bar charts (**graph hbar**; also in **graph hbox** and some other turned-sideways graphics) the horizontal axis is called the *y* axis, contrary to usual convention. Thus **ytitle("Weighted percent")** specifies a title that goes along the bottom. Stata does not recognize an *x* axis in these charts, although **l1title()** could specify a title that would go along the left-hand side.

Figure 4.4 compared only two numbers, 76% among college graduates and 64% among others. No one needs to draw a graph to make such a simple comparison. This bar-chart approach readily extends to more complicated comparisons, however. Previous surveys have found pervasive partisan divisions across many questions involving climate change. Might those exist even for this factual question about Arctic sea ice? A three-variable bar chart in Figure 4.5 gives the percent of correct answers broken down by college education and respondent's political identification (variable *party*).

```
. graph hbar (mean) warmiceQ100 [aw = censuswt],
  over(college) over(party)
  blabel(bar, format(%3.0f)) ytitle("Weighted percent")
  title("Arctic ice declined, by college and political party"
    , size(medium))
```

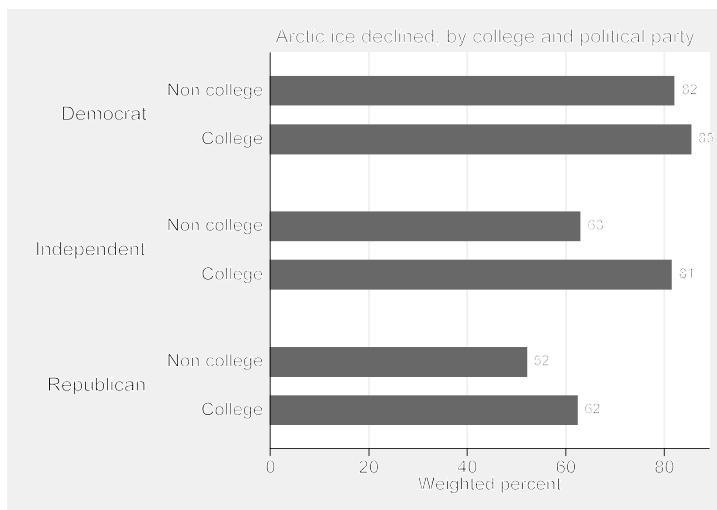


Figure 4.5

Within each *party* group we see a college-education effect, and within each level of education we see a partisan difference as well.

Figure 4.5 cannot tell us whether the differences are statistically significant. Answering that question requires statistical modeling tools introduced in Chapter 9. As a preview, however, the following weighted logit regression model confirms that both *college* and *party* have statistically significant effects, positive in the case of *college* (0 = non-college, 1 = college grad) and negative in the case of *party* (1 = Democrat, 2 = Independent, 3 = Republican). *Party* is a stronger predictor than *college*.

```
. svy: logit warmiceQ college party
(running logit on estimation sample)

Survey: Logistic regression

Number of strata     =           1
Number of PSUs        =         501
                                         Number of obs      =      501
                                         Population size  = 500.96122
                                         Design df        =      500
                                         F(  2,    499)   =     16.07
                                         Prob > F        =     0.0000
```

warmiceQ	Coef.	Linearized Std. Err.	t	P> t	[95% Conf. Interval]
college	.4634607	.2264922	2.05	0.041	.018467 .9084544
party	-.6669759	.1268445	-5.26	0.000	-.9161897 -.4177621
_cons	2.058491	.3162993	6.51	0.000	1.437052 2.679931

Chapter 9 will return to this example, applying a statistical method (multinomial logit) that can model the predictors of each separate answer to the ice-area question.

Summary Statistics and Tables

The **summarize** command finds simple descriptive statistics such as medians, means and standard deviations of measurement variables. Flexible arrangements of summary statistics by other variables are available through the command **tabstat**. For categorical or ordinal variables, **tabulate** obtains frequency distribution tables, cross-tabulations, assorted tests, and measures of association. **tabulate** can also construct one- or two-way tables of means and standard deviations across categories of other variables. A general table-making command, **table**, produces up to seven-way tables in which the cells contain statistics such as frequencies, sums, means, or medians. Finally, we review further one-variable procedures including normality tests, transformations and displays for exploratory data analysis (EDA). Most of the analyses covered in this chapter can be accomplished either through the commands shown or through menu selections under Statistics > Summaries, tables & tests .

In addition to such general-purpose analyses, Stata provides many tables of particular interest to epidemiologists. These are not described in this chapter, but can be viewed by typing **help epitab**. Selvin (2004) introduces the topic.

Example Commands

- **summarize y1 y2 y3**
Calculates simple summary statistics (means, standard deviations, minimum and maximum values, and numbers of observations) for the variables listed.
- **summarize y1 y2 y3, detail**
Obtains detailed summary statistics including percentiles, median, mean, standard deviation, variance, skewness and kurtosis.
- **summarize y1 if x1 > 3 & !missing(x2)**
Finds summary statistics for *y1* using only those observations for which variable *x1* is greater than 3, and *x2* is not missing.
- **summarize y1 [fweight = w], detail**
Calculates detailed summary statistics for *y1* using the frequency weights in variable *w*.
- **tabstat y1, stats(mean sd skewness kurtosis n)**
Calculates only the specified summary statistics for variable *y1*.

- . **tabstat y1, stats(min p5 p25 p50 p75 p95 max) by(x1)**
Calculates the specified summary statistics (minimum, 5th percentile, 25th percentile, etc.) for measurement variable *y1*, within categories of *x1*.
- . **tabulate x1**
Displays a frequency distribution table for all nonmissing values of variable *x1*.
- . **tabulate x1, sort miss**
Displays a frequency distribution of *x1*, including the missing values. Rows (values) are sorted from most to least frequent.
- . **tab1 x1 x2 x3 x4**
Displays a series of frequency distribution tables, one for each of the variables listed.
- . **tabulate x1 x2**
Displays a two-variable cross-tabulation with *x1* as the row variable, and *x2* as the columns.
- . **tabulate x1 x2, chi2nof column**
Produces a cross-tabulation and Pearson χ^2 test of independence. Does not show cell frequencies, but instead gives the column percentages in each cell.
- . **tabulate x1 x2, missing row all**
Produces a cross-tabulation that includes missing values in the table and in the calculation of percentages. Calculates all available statistics (Pearson and likelihood χ^2 , Cramér's *V*, Goodman and Kruskal's gamma, and Kendall's τ_b).
- . **tab2 x1 x2 x3 x4**
Performs all possible two-way cross-tabulations of the listed variables.
- . **tabulate x1, summ(y)**
Produces a one-way table showing the mean, standard deviation, and frequency of *y* values within each category of *x1*.
- . **tabulate x1 x2, summ(y) means**
Produces a two-way table showing the mean of *y* at each combination of *x1* and *x2* values.
- . **by x3, sort: tabulate x1 x2, exact**
Creates a three-way cross-tabulation, with subtables for *x1* (row) by *x2* (column) at each value of *x3*. Calculates Fisher's exact test for each subtable. **by varname, sort:** works as a prefix for almost any Stata command where it makes sense. The **sort** option is unnecessary if the data already are sorted on *varname*.
- . **table y x2 x3, by(x4 x5) contents(freq)**
Creates a five-way cross-tabulation, of *y* (row) by *x2* (column) by *x3* (supercolumn), by *x4* (superrow 1) by *x5* (superrow 2). Cells contain frequencies.
- . **table x1 x2, contents(mean y1 median y2)**
Creates a two-way table of *x1* (row) by *x2* (column). Cells contain the mean of *y1* and the median of *y2*.

. svy: tab y, percent ci

Using survey-weighted data (as declared by **svyset**), obtains a one-way table of percentages for variable *y*, with 95% confidence intervals. Type **help svy tab** for more survey table options. Chapter 14 introduces survey data and analysis.

. svy: tab y x, column percent

Using survey-weighted data, obtains a two-way cross-tabulation of row variable *y* against column variable *x*, with adjusted χ^2 test of independence. Cells contain weighted column percentages.

Summary Statistics for Measurement Variables

Dataset *electricity.dta* contains information on electricity consumption in U.S. states, from the California Energy Commission (2012).

```
. use C:\data\electricity.dta, clear
. describe

Contains data from C:\data\electricity.dta
obs:                      51                               US states electricity use 2010
vars:                      7                               (CA Energy Commission)
size:                   1,785                             13 Mar 2012 10:28

variable   storage      display      value      variable
          name       type        format     label      label
state       str21      %21s
stateab     str2       %9s
region4     byte       %9.0g      reg4
region9     byte       %12.0g     reg9
pop         long       %8.0g
electric    long       %8.0g
elcap       int        %8.0g

Sorted by: state
```

To find the mean and standard deviation of per capita electricity use (*elcap*), type

. summarize elcap

Variable	Obs	Mean	Std. Dev.	Min	Max
elcap	51	13318.43	4139.328	6721	27457

This table also gives the number of nonmissing observations and the variable's minimum and maximum values. If we had simply typed **summarize** with no variable list, we would obtain means and standard deviations for every numeric variable in the dataset.

To see more detailed summary statistics, type

. summarize elcap, detail

Per capita electricity use, kwh				
	Percentiles	Smallest		
1%	6721	6721		
5%	7434	7363		
10%	8286	7434	Obs	51
25%	10359	7467	Sum of wgt.	51
50%	13388		Mean	13318.43
		Largest	Std. Dev.	4139.328
75%	16117	19477		
90%	17903	19896	Variance	1.71e+07
95%	19896	21590	Skewness	.7643711
99%	27457	27457	Kurtosis	4.161063

This **summarize**, **detail** output includes basic statistics plus the following:

Percentiles: Notably the first quartile (25th percentile = 10,359), median (50th percentile = 13,388), and third quartile (75th percentile = 16,117). Because many samples do not divide evenly into quarters or other standard fractions, these percentiles are approximations.

Four smallest and four largest values, where outliers might show up.

Sum of weights: the **summarize** command permits frequency weights or **fweight**. For explanations see **help weight**.

Variance: Standard deviation squared (more properly, standard deviation equals the square root of variance).

Skewness: The direction and degree of asymmetry. A perfectly symmetrical distribution has skewness = 0. Positive skew (heavier right tail) results in skewness > 0; negative skew (heavier left tail) results in skewness < 0.

Kurtosis: Tail weight. A normal (Gaussian) distribution is symmetrical with kurtosis = 3. If a symmetrical distribution has heavier-than-normal tails (is sharply peaked), then kurtosis > 3. Kurtosis < 3 indicates lighter-than-normal tails.

The **tabstat** command provides a more flexible alternative to **summarize**. We can specify just which summary statistics we want to see. For example,

. tabstat elcap, stats(mean min max)			
variable	mean	min	max
elcap	13318.43	6721	27457

With a **by(varname)** option, **tabstat** constructs a table containing summary statistics for each value of *varname*. The following example gives means, minimum and maximum for per capita electricity use, separately for each of four U.S. census regions. Electricity use is relatively low in the Northeast, and much higher in the Midwest and South.

```
. tabstat elcap, stats(mean min max) by(region4)
```

**Summary for variables: elcap
by categories of: region4 (Census Region (4))**

region4	mean	min	max
Northeast	8746	7434	11759
Midwest	14151.5	10516	19477
South	16001.06	11343	21590
West	12206.92	6721	27457
Total	13318.43	6721	27457

In addition to **mean**, **min** or **max**, other statistics available for the **stats()** option of **tabstat** include the same set listed earlier for **collapse** or **graph bar** (such as **count**, **sum**, **max**, **min**, **variance**, **sd**, and **p1** through **p99** for percentiles). Further **tabstat** options give control over the table layout and labeling. Type **help tabstat** to see a complete list.

The statistics produced by **summarize** or **tabstat** describe the sample at hand. For some purposes, although probably not with U.S. states data, we might want to construct confidence intervals suggesting inferences about a larger population. As an illustration, obtaining a 99% confidence interval for the mean of *elcap*:

```
. ci elcap, level(99)
```

Variable	Obs	Mean	Std. Err.	[99% Conf. Interval]
elcap	51	13318.43	579.6218	11766.32 14870.54

Accepting for the moment these 51 states (with the District of Columbia) as a sample, we could be 99% confident that the population mean lies somewhere in the interval from 11,766 to 14,870 kWh per person. More precisely, over many random samples, intervals constructed in this manner should contain the true population mean about 95% of the time. The **level(99)** option specified a 99% confidence interval. If we omit this option, **ci** defaults to a 95% confidence interval.

Other options allow **ci** to calculate exact confidence intervals for variables that follow binomial or Poisson distributions. A related command, **cii**, calculates normal, binomial or Poisson confidence intervals directly from summary statistics, such as we might encounter in a published article. It does not require the raw data. Type **help ci** for details about both of these useful commands.

Exploratory Data Analysis

Statistician John Tukey assembled a toolkit of old and new methods for exploratory data analysis (EDA), which involves analyzing data in an exploratory and skeptical way without making unneeded assumptions (see Tukey 1977; also Hoaglin, Mosteller and Tukey 1983, 1985). Box plots, introduced in Chapter 3, are one the most popular EDA methods. Another is the stem-and-leaf display, a graphical arrangement of ordered data values in which initial digits form the stems and following digits for each observation make up the leaves.

```
. stem elcap
STEM-and-leaf plot for elcap (Per capita electricity use, kwh)

 6*** 721
 7*** 363,434,467,952
 8*** 286,514,591,696,982,985
 9*** 
10*** 106,359,516,739
11*** 253,343,395,759
12*** 077,159,379,497,845,904
13*** 388,557,916,992
14*** 179,263,325,345,475,489,578
15*** 048,568
16*** 117,293,315,519,793
17*** 290,293,903
18*** 852
19*** 477,896
20*** 
21*** 590
22*** 
23*** 
24*** 
25*** 
26*** 
27*** 457
```

In this display the lowest per capita electricity value, 6,721 (California), appears as a 721 leaf on the 6*** stem. The highest, 27,457 (Wyoming), appears as a 457 leaf on the 27*** stem. **stem** automatically chooses values for the stems, but can override this with a **lines()** option. Type **help stem** for information about this option and others.

Letter-value displays (**lv**) use order statistics to dissect a distribution.

#	51	Per capita electricity use, kwh			
M	26	13388		spread	pseudosigma
F	13.5	10437.5	13140	5405	4131.039
E	7	8514	12903.5	8779	3894.835
D	4	7467	13472	12010	4098.322
C	2.5	7398.5	14070.75	13344.5	3866.579
B	1.5	7042	15782.75	17481.5	4369.45
	1	6721	17089	20736	4689.655
				# below	# above
inner fence		2330	23950	0	1
outer fence		-5777.5	32057.5	0	0

M denotes the median, and F the “fourths” (quartiles, using a different approximation than the quartile approximation used by **summarize**, **detail** and **tabsum**). E , D , C , . . . denote cutoff points such that roughly 1/8, 1/16, 1/32, . . . of the distribution remains outside in the tails. The second column of numbers gives the depth or distance from the nearest extreme, for each letter value. Within the center box, the middle column gives midsummaries, which are averages of the two letter values. If midsummaries drift away from the median, as they do for *elcap*, this tells us that the distribution becomes progressively more skewed as we move farther out into the tails. The spreads are differences between pairs of letter values. For instance, the spread between Fs equals the approximate interquartile range. Finally, pseudosigmas in the right-hand column estimate what the standard deviation should be if these letter values described a Gaussian

population. The F pseudosigma, sometimes called a pseudo standard deviation (*PSD*), provides a simple and outlier-resistant check for approximate normality in symmetrical distributions:

1. Comparing mean with median diagnoses overall skew:

mean > median	positive skew
mean = median	symmetry
mean < median	negative skew

2. If the mean and median are similar, indicating symmetry, then a comparison between standard deviation and *PSD* helps to evaluate tail normality:

standard deviation > <i>PSD</i>	heavier-than-normal tails
standard deviation = <i>PSD</i>	normal tails
standard deviation < <i>PSD</i>	lighter-than-normal tails

Let F_1 and F_3 denote 1st and 3rd quartiles (approximate 25th and 75th percentiles). Then the interquartile range, *IQR*, equals $F_3 - F_1$, and $PSD = IQR / 1.349$.

IV also identifies mild and severe outliers, if any exist (there is only one mild outlier in the *elcap* distribution). We call an x value a “mild outlier” when it lies outside the inner fence, but not outside the outer fence:

$$F_1 - 3IQR \leq x < F_1 - 1.5IQR \quad \text{or} \quad F_3 + 1.5IQR < x \leq F_3 + 3IQR$$

The value of x is a “severe outlier” if it lies outside the outer fence:

$$x < F_1 - 3IQR \quad \text{or} \quad x > F_3 + 3IQR$$

IV gives these cutoffs and the number of outliers of each type. Severe outliers, values beyond the outer fences, occur sparsely (about two per million) in normal populations. Monte Carlo simulations suggest that the presence of any severe outliers in samples of $n = 15$ to about 20,000 should be sufficient evidence to reject a normality hypothesis at $\alpha = .05$ (Hamilton 1992b). Severe outliers create problems for many statistical techniques.

summarize, **stem** and **IV** all confirm that *lived* has a positively skewed sample distribution, not resembling a theoretical normal curve. The next section introduces more formal normality tests, and transformations that can reduce a variable’s skew.

Normality Tests and Transformations

Many statistical procedures work best when applied to variables that follow normal distributions. The preceding section described exploratory methods to check for approximate normality, extending the graphical tools (histograms, box plots, symmetry plots and quantile–normal plots) presented in Chapter 3. A skewness–kurtosis test, making use of the skewness and kurtosis statistics shown by **summarize**, **detail**, can more formally evaluate the null hypothesis that the sample at hand came from a normally-distributed population.

. **sktest elcap**

Variable	Skewness/Kurtosis tests for Normality				
	Obs	Pr(Skewness)	Pr(Kurtosis)	adj chi2(2)	Prob>chi2
elcap	51	0.0223	0.0723	7.49	0.0236

sktest here rejects normality: *elcap* appears significantly nonnormal in skewness ($p = .0223$), although not in kurtosis ($p = .0723$), and in both statistics considered jointly ($p = .0236$).

Other normality tests include Shapiro–Wilk W (**swilk**) and Shapiro–Francia W' (**sfrancia**) methods (type **help sktest**). A Stata module to perform Doornik–Hansen omnibus tests for univariate/multivariate normality is available online (type **findit omninorm**).

Nonlinear transformations such as square roots and logarithms are often employed to change distributions' shapes, with the aim of making skewed distributions more symmetrical and perhaps more nearly normal. Transformations might also help linearize relationships between variables (Chapters 7 and 8). Table 5.1 shows a progression called the ladder of powers (Tukey 1977) that provides guidance for choosing transformations to change distributional shape. The variable *lived* exhibits mild positive skew, so its square root might be more symmetrical. We could create a new variable equal to the square root of *elcap* by typing

```
. generate srelcap = elcap ^ .5
```

Instead of *elcap* $^.5$, we could equally well have written **sqrt(elcap)**.

Logarithms are another transformation that can reduce positive skew. To generate a new variable equal to the natural (base e) logarithm of *elcap*, type

```
. generate logelcap = ln(elcap)
```

In the ladder of powers and related transformation schemes such as Box–Cox, logarithms take the place of a “0” power. Their effect on distribution shape is intermediate between $.5$ (square root) and $-.5$ (reciprocal root) transformations.

Table 5.1: Ladder of Powers

Transformation	Formula	Effect
cube	<i>new</i> = <i>old</i> 3	reduce severe negative skew
square	<i>new</i> = <i>old</i> 2	reduce mild negative skew
raw	<i>old</i>	no change (raw data)
square root	<i>new</i> = <i>old</i> $^.5$	reduce mild positive skew
\log_e (or \log_{10})	<i>new</i> = $\ln(\text{old})$	reduce positive skew
negative reciprocal root	<i>new</i> = $-(\text{old} ^{-0.5})$	reduce severe positive skew
negative reciprocal	<i>new</i> = $-(\text{old} ^{-1})$	reduce very severe positive skew
negative reciprocal square	<i>new</i> = $-(\text{old} ^{-2})$	"
negative reciprocal cube	<i>new</i> = $-(\text{old} ^{-3})$	"

We take negatives of the result after raising to a power less than zero, to preserve the original order — the highest value of *old* becomes transformed into the highest value of *new*, and so forth. When *old* itself contains negative or zero values, it is necessary to add a constant before

transformation. For example, if *arrests* measures the number of times a person has been arrested (0 for many people), then a suitable log transformation could be

```
. generate logarrest = ln(arrests + 1)
```

The **ladder** command combines the ladder of powers with **sktest** for normality. It tries each power on the ladder, and reports whether the result is significantly nonnormal. This can be illustrated using the positively skewed variable *elcap*, per capita electricity consumption, from *electricity.dta*.

```
. ladder elcap
```

Transformation	formula	chi2(2)	P(chi2)
cubic	<i>elcap</i> ³	44.12	0.000
square	<i>elcap</i> ²	26.24	0.000
identity	<i>elcap</i>	7.49	0.024
square root	sqrt(<i>elcap</i>)	1.21	0.547
log	log(<i>elcap</i>)	0.26	0.879
1/(square root)	1/sqrt(<i>elcap</i>)	2.36	0.307
inverse	1/ <i>elcap</i>	4.87	0.088
1/square	1/(<i>elcap</i> ²)	10.67	0.005
1/cubic	1/(<i>elcap</i> ³)	17.51	0.000

Square root, log, inverse square root and inverse transformations all yield distributions that are not significantly different from normal. In this respect they offer improvements compared with the raw data or identity transformation, with is significantly nonnormal ($p = .024$). It appears that logarithms offer the best choice for a normalizing transformation. Figure 5.1, produced by the **gladder** command, visually supports this conclusion by comparing histograms of each transformation to normal curves.

```
. gladder elcap
```

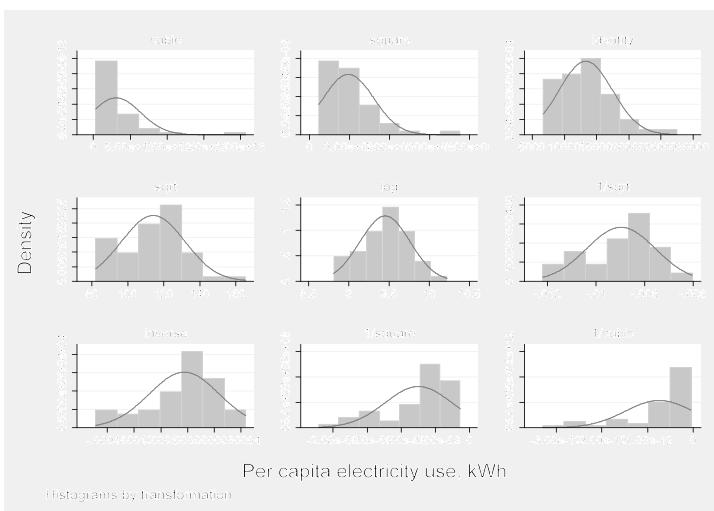


Figure 5.1

Figure 5.2 shows a corresponding set of quantile–normal plots for these ladder of powers transformations, obtained by the quantile ladder command **qladder**. (Type **help ladder** for information about **ladder**, **gladder** and **qladder**.) To make the tiny plots more readable in the example below, we scale the labels and marker symbols up by 25% with the **scale(1.25)** option. The axis labels (which would be unreadable and crowded) are suppressed by the options **ylabel(none)** **xlabel(none)**.

```
. qladder elcap, scale(1.25) ylabel(none) xlabel(none)
```

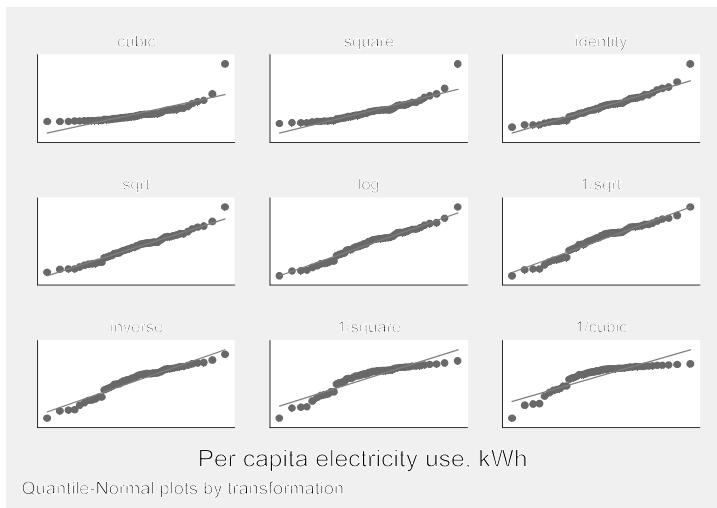


Figure 5.2

An alternative technique called Box–Cox transformation offers finer gradations between transformations and automates the choice among them (easier for the analyst, but not always a good thing). The command **bcskew0** finds a value of λ (lambda) for the transformations

$$y^{(\lambda)} = \{y^\lambda - 1\} / \lambda \quad \lambda > 0 \text{ or } \lambda < 0$$

or

$$y^{(\lambda)} = \ln(y) \quad \lambda = 0$$

such that $y^{(\lambda)}$ has approximately zero skewness. Applying this to *elcap*, we obtain the transformed variable *belcap*:

```
. bcskew0 belcap = elcap, level(95)
```

Transform	L	[95% Conf. Interval]	Skewness
$(elcap^{.145} - 1) / (.145)$.1451061	-.8268476 .8784035	3.75e-06

That is, $belcap = (elcap^{.145} - 1) / (.145)$ is the transformation that comes closest to symmetry, as defined by the skewness statistic. The Box–Cox parameter $\lambda = .145$ is not far from our ladder-of-powers choice, logarithms (which take the place of the 0 power). The confidence interval for λ includes 0 (logarithm) but not 1 (identity or no change):

$$-.827 < \lambda < .878$$

Chapter 8 describes a Box–Cox approach to regression modeling.

Frequency Tables and Two-Way Cross-Tabulations

The summary statistics, graphs and transformations described above apply mainly to measurement variables. Categorical variables require different approaches, often starting with simple one- or two-way tables. For examples using such tables we return to the Granite State Poll data, *Granite2011_6.dta*. One question (*trackus*) asked whether people think the U.S. is headed in the right direction, or is on the wrong track. Although this question appears vague and strangely worded it has a long tradition among pollsters, having been used successfully on hundreds of surveys as a barometer of public mood. A majority on this New Hampshire poll expressed pessimism:

. tabulate trackus

US right direction or wrong track	Freq.	Percent	Cum.
Right direction	176	37.29	37.29
wrong track	296	62.71	100.00

Total	472	100.00	
-------	-----	--------	--

tabulate can produce frequency distributions for variables that have thousands of values. To construct a manageable frequency distribution table for a variable with many values, however, you might first want to group those values by applying **generate** with its **recode** or **autocode** options (see Chapter 2 or **help generate**).

tabulate followed by two variable names creates a two-way cross-tabulation. For example, here is a cross-tabulation of *trackus* by *educ* (respondent's level of education):

. tabulate educ trackus

Highest degree completed	US right direction or wrong track		Total
	Right dir	Wrong tra	
HS or less	36	71	107
Tech/some coll	34	76	110
College grad	49	93	142
Postgrad	56	52	108
Total	175	292	467

The first-named variable forms the rows, and the second forms columns in the resulting table. We see that 71 of the 107 respondents with high-school education or less believe the U.S. is on the wrong track.

Are *trackus* opinions related to education? To find out we can run a χ^2 (chi-squared) test and examine row percentages, because *educ*, which defines the rows, is our independent variable here. The **row** option asks for row percentages; **nof** means no frequencies should be shown.

. tabulate educ trackus, row nof chi2

Highest degree completed	US right direction or wrong track		Total
	Right dir	Wrong tra	
HS or less	33.64	66.36	100.00
Tech/some coll	30.91	69.09	100.00
College grad	34.51	65.49	100.00
Postgrad	51.85	48.15	100.00
Total	37.47	62.53	100.00

Pearson chi2(3) = 12.7549 Pr = 0.005

Sixty-nine percent of respondents with technical school or some college believe the U.S. is on the wrong track. Those with postgraduate degrees appear more optimistic; only 48% hold this view. Based on this sample, we can reject the null hypothesis that *educ* and *trackus* are unrelated in the New Hampshire population ($\chi^2 = 12.75, p = .005$).

tabulate has a number of options that are useful with two-way tables. These include alternative tests (Fisher's exact test; likelihood-ratio χ^2) and measures of association (Goodman and Kruskal's γ (gamma), Kendall's τ_b (tau-b) and Cramér's V). The option **missing** requests that missing values be included among the rows or columns of the table. **tabulate** can also save frequencies and variables names as matrices. Type **help tabulate** for list of options with details.

Occasionally we might need to re-analyze a published table, without access to the original raw data. A special command, **tabi** (immediate tabulation), accomplishes this. Type the cell frequencies on the command line, with table rows separated by "\". For illustration, here is how **tabi** could reproduce the previous cross-tabulation directly from the four cell frequencies without need for any dataset:

. **tabi** 36 71 \ 34 76 \ 49 93 \ 56 52

row	col		Total
	1	2	
1	36	71	107
2	34	76	110
3	49	93	142
4	56	52	108
Total	175	292	467

Pearson chi2(3) = 12.7549 Pr = 0.005

And here is the same analysis showing row percentages and χ^2 test:

. **tabi** 36 71 \ 34 76 \ 49 93 \ 56 52, row nof chi2

row	col		Total
	1	2	
1	33.64	66.36	100.00
2	30.91	69.09	100.00
3	34.51	65.49	100.00
4	51.85	48.15	100.00
Total	37.47	62.53	100.00

Pearson chi2(3) = 12.7549 Pr = 0.005

Unlike **tabulate**, **tabi** does not require or refer to any data in memory. By adding the **replace** option, however, we can ask **tabi** to replace whatever data are in memory with the new cross-tabulation. Statistical options (**chi2**, **exact**, **nofreq** and so forth) work the same for **tabi** as they do with **tabulate**; see **help tabulate twoway**.

None of the examples in this section so far involve weighting. As described in Chapter 4, survey researchers often do apply weights, carefully calculated to make sample results more representative of a target population. The variable *censuswt* provides such weights for the Granite State Poll. Earlier, we used a **svyset** command to declare these as probability weights.

```
. svyset [pw = censuswt]
```

Commands with the **svy:** prefix will apply the **svyset** weights automatically; for other commands the weights are ignored. Here are weighted versions of the one and two-way tables above.

```
. svy: tab trackus  
(running tabulate on estimation sample)
```

Number of strata	=	1	Number of obs	=	472
Number of PSUs	=	472	Population size	=	474.80568
			Design df	=	471

us right direction or wrong track	proportions
Right di	.3696
wrong tr	.6304
Total	1

Key: proportions = cell proportions

```
. svy: tab educ trackus, row percent  
(running tabulate on estimation sample)
```

Number of strata	=	1	Number of obs	=	467
Number of PSUs	=	467	Population size	=	469.25491
			Design df	=	466

Highest degree completed	us right direction or wrong track		
	Right di	wrong tr	Total
HS or le	34.33	65.67	100
Tech/som	24.5	75.5	100
College	36.41	63.59	100
Postgrad	53.41	46.59	100
Total	37.26	62.74	100

Key: row percentages

Pearson:
Uncorrected chi2(3) = 21.3629
Design-based F(2.99, 1394.32)= 5.9918 P = 0.0005

In the weighted table we see an even greater difference in pessimism between respondents with technical school or some college (75.5% wrong track) and postgraduate education (46.6% wrong track). Design-based *F* tests provide the survey counterpart to an unweighted table's chi-squared test. This *F* test also agrees that the *educ/trackus* relationship is statistically significant ($p = .0005$).

Multiple Tables and Multi-Way Cross-Tabulations

With surveys and other large datasets, we sometimes need frequency distributions of many different variables. Instead of asking for each table separately, for example by typing **tabulate tparty**, then **tabulate obama**, and finally **tabulate trackus**, we could simply use another specialized command, **tab1**:

```
. tab1 tparty obama trackus
```

Or, to produce one-way frequency tables for each variable from *tparty* through *trackus* in this dataset (the maximum is 30 variables at one time), type

```
. tab1 tparty-obama
```

Similarly, **tab2** creates multiple two-way tables. For example, the following command cross-tabulates every two-way combination of the listed variables:

```
. tab2 tparty obama trackus
```

tab1 and **tab2** offer the same options as **tabulate**.

To form multi-way contingency tables, it is possible to use **tabulate** with a **by** prefix. For example, here is a simple two-way cross-tabulation of whether the survey respondents voted for Obama in 2008, by whether or not they graduated from college.

```
. tab obama college, col nof chi
```

Voted for Obama in 2008	College graduate		Total
	Non colle	College	
No	61.44	41.45	50.68
Yes	38.56	58.55	49.32
Total	100.00	100.00	100.00

Pearson chi2(1) = 20.2966 Pr = 0.000

One way to make a three-way cross-tabulation of the *obama/college* relationship by gender is to use **sort** and the **by:** prefix. This produces two-way tables with the same layout as above, but separately for males and females.

```
. sort sex
. by sex: tab obama college, col nof chi
```

-> sex = Male

Voted for Obama in 2008	College graduate		Total
	Non colle	College	
No	69.81	44.88	56.22
Yes	30.19	55.12	43.78
Total	100.00	100.00	100.00

Pearson chi2(1) = 14.5888 Pr = 0.000

-> sex = Female

Voted for Obama in 2008	College graduate		Total
	Non colle	College	
No	54.62	38.51	46.04
Yes	45.38	61.49	53.96
Total	100.00	100.00	100.00

Pearson chi2(1) = 7.2227 Pr = 0.007

The *obama/college* relationship is significant and has the same direction in both sub-tables, but appears somewhat stronger among men (where college makes a 25-point difference, 30.19 to 55.12%) than women (a 16-point difference, 45.38 to 61.79%).

This approach can be extended to tabulations of greater complexity. To get a four-way cross-tabulation of *obama* by *college*, with separate sub-tables for married and unmarried men and women, we could type the commands (results not shown):

```
. sort sex married
. by sex married: tab obama college, col nof chi
```

Such multi-way tables divide the data into increasingly small subsamples, where variations become more erratic.

An alternative way to produce multi-way tables, if we do not need percentages or statistical tests, is through Stata's general table-making command, **table**. This versatile command has many options, only a few of which are illustrated here. To construct a two-way table of *obama* by *college*, with frequencies in each cell, type

```
. table obama college, contents(freq)
```

Voted for Obama in 2008	College graduate	
	Non college	College
No	145	114
Yes	91	161

If we specify a third categorical variable, it forms the supercolumns of a three-way table:

```
. table obama college sex, contents(freq)
```

Voted for Obama in 2008	Gender and College graduate			
	Male	Non college	Female	College
No	74	57	71	57
Yes	32	70	59	91

More complicated tables require the `by()` option, which allows up to four superrow variables. `table` thus can produce up to seven-way tables: one row, one column, one supercolumn, and up to four superrows. Here is a four-way example:

```
. table obama college sex, contents(freq) by(married)
```

Respondent married and Voted for Obama in 2008	Gender and College graduate			
	Male	Non college	Female	College
No	34	12	35	22
Yes	15	22	39	38
Yes	40	45	36	35
Yes	17	48	20	53

The `table` examples above all place frequencies in the cells, but `table` permits statistical summaries as well. Here is a four-way table, `obama` × `college` × `sex` × `married`, in which each cell contains the mean `age` for that combination of characteristics. For example, we see that the 34 non-college, unmarried men who did not vote for Obama have a mean age of 46.6 years.

```
. table obama college sex, contents(mean age) by(married)
```

Respondent married and Voted for Obama in 2008	Gender and College graduate			
	Male	Non college	Female	College
No	46.63636	46.91667	59.69697	58.22222
Yes	53	53.45454	62.37838	61.78378
Yes	56.075	55.92857	58.2	52.35484
Yes	59	55.87234	53.21053	53.80769

The `contents()` option of `table` specifies what statistics the table's cells contain. The choices include not just frequency or mean, but also standard deviation, minimum, maximum, median, range, percentiles and other summaries. Type `help table` for a full list. The next section illustrates some other possibilities for summary statistics within tables.

Tables of Means, Medians and Other Summary Statistics

tabulate produces tables of means and standard deviations within categories of the tabulated variable. For the remaining examples in this chapter, we return to the data on electricity consumption in U.S. states. **tabulate** gives us one way to view summary statistics of per capita electricity consumption (*elcap*) for each of the 9 U.S. census divisions (*region9*):

```
. tabulate region9, summ(elcap)
```

Census Division (9)	Summary of Per capita electricity use, kwh		
	Mean	Std. Dev.	Freq.
New Engla	8417.1667	532.95419	6
Mid Atlan	9403.6667	2176.4139	3
E N Centr	12726.2	2274.5595	5
W N Centr	15169.571	2172.0833	7
S Atlanti	15011.889	2810.0798	9
E S Centr	17948.25	2475.1953	4
W S Centr	16279.5	1965.8288	4
Mountain	13877.5	5723.3327	8
Pacific	9534	3073.2846	5
Total	13318.431	4139.3277	51

We could also use **tabulate** to form a two-way table of means, as in this contrived example using the 9 census divisions and 4 census regions:

```
. tabulate region9 region4, summ(elcap) mean
```

Means of Per capita electricity use, kwh

Census Division (9)	Census Region (4)				Total
	Northeast	Midwest	South	West	
New Engla	8417.1667	.	.	.	8417.1667
Mid Atlan	9403.6667	.	.	.	9403.6667
E N Centr	.	12726.2	.	.	12726.2
W N Centr	.	15169.571	.	.	15169.571
S Atlanti	.	.	15011.889	.	15011.889
E S Centr	.	.	17948.25	.	17948.25
W S Centr	.	.	16279.5	.	16279.5
Mountain	.	.	.	13877.5	13877.5
Pacific	.	.	.	9534	9534
Total	8746	14151.5	16001.059	12206.923	13318.431

The **means** option above called for a table containing only means. Otherwise we get a bulkier table with means, standard deviations and frequencies within each cell.

The more flexible **table** command can build up to seven-way tables containing means, standard deviations, sums, medians or other statistics. For illustration, here is a one-way table showing mean and standard deviation of per capita electricity use and also the median and interquartile range of population, within each census division.

```
. table region9, contents (mean elcap sd elcap median pop iqr pop)
```

Census Division (9)	mean(elcap)	sd(elcap)	med(pop)	iqr(pop)
New England	8417.17	532.9542	1322	2521
Mid Atlantic	9403.67	2176.414	12702	10586
E N Central	12726.2	2274.56	9884	5053
W N Central	15169.6	2172.083	2853	4490
S Atlantic	15011.9	2810.08	5774	7682
E S Central	17948.3	2475.195	4559.5	1910
W S Central	16279.5	1965.829	4142	11506
Mountain	13877.5	5723.333	2380	2618
Pacific	9534	3073.285	3831	5365

Mean per capita electricity use varies by a factor of two, from the low of 8,417 kWh in New England to a high of 17,948 kWh in the West South Central division (which includes the oil-producing states of Texas, Louisiana and Oklahoma). The greatest variation, on the other hand, occurs among the Mountain states, where the standard deviation (5,723 kWh) is ten times that of New England (533 kWh).

The **contents()** option with **table** specifies what statistics, for what variables, each cell should contain. The possible statistics also include minimum, maximum, sum, percentiles and several types of standard error. See **help table** for the list.

Using Frequency Weights

summarize, **tabulate**, **table** and related commands can be used with frequency weights that indicate the number of replicated observations. For example, here are the mean and other statistics for per capita electricity use across all U.S. states.

```
. summ elcap
```

Variable	Obs	Mean	Std. Dev.	Min	Max
elcap	51	13318.43	4139.328	6721	27457

This mean, 13,318 kWh, tells us the average electricity across the 51 states (including District of Columbia) — counting each state as one unit. Wyoming has the smallest population (564 thousand) and the highest per capita electricity consumption (27,457 kWh). California has the largest population (37 million) and the lowest per capita electricity consumption (6,721 kWh). Each has equal weight in this 51-state mean. To see the per capita mean for the U.S. as a whole, however, we need to weight by population.

```
. summ elcap [fweight = pop]
```

Variable	Obs	Mean	Std. Dev.	Min	Max
elcap	308746	12112.69	3519.441	6721	27457

The population-weighted mean (12,112 kWh) is lower than the mean of the 51 states (13,318 kWh) because more people live in relatively low-consuming states such as California and New York than in high-consuming states such as Wyoming and Kentucky (Figure 5.3).

```
. graph twoway scatter elcap pop, mlabel(state)
```

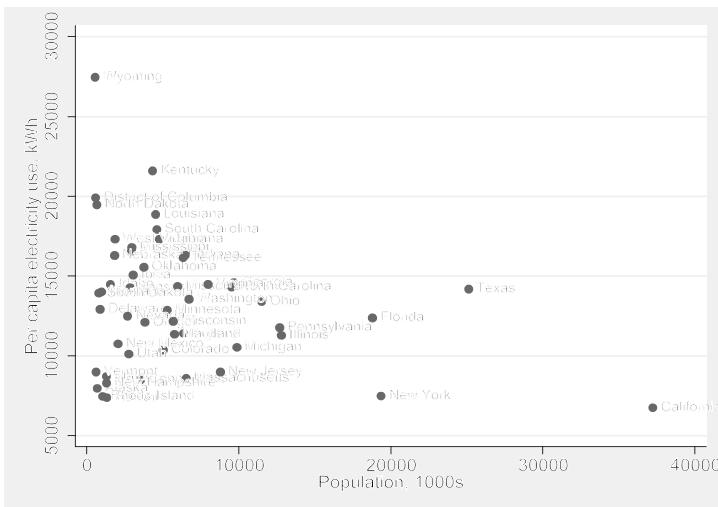


Figure 5.3

The population-weighted mean, unlike the unweighted mean, can be interpreted as a mean for the 309 million people in the U.S. Note, however, that we could not make similar statements for the weighted standard deviation, minimum or maximum. Apart from the mean, most individual-level statistics cannot be calculated simply by weighting data that already are aggregated. Thus, we need to use weights with caution. They might make sense in the context of one particular analysis, but seldom do for the dataset as a whole, when many different kinds of analyses are needed.

Frequency weights operate in a similar way with both **tabulate** and **table**. The following **table** command calculates population-weighted means for each census division. Now that we take its much larger population into account, we see that the Pacific division has lower mean electricity consumption than New England.

```
. table region9 [fweight = pop], contents(mean elcap) row
```

Census Division (9)	mean(elcap)
New England	8486.42
Mid Atlantic	9127.38
E N Central	12444.7
W N Central	14374.9
S Atlantic	13891.4
E S Central	17819.2
W S Central	15092.9
Mountain	11816.9
Pacific	8089.04
Total	12112.7

The **row** option called for a final row summarizing the table as a whole. The overall mean in this table (12,112.7 kWh) is the same as that found earlier by **summarize**.

ANOVA and Other Comparison Methods

Analysis of variance (ANOVA) encompasses a set of methods for testing hypotheses about differences between means. Its applications range from simple analyses where we compare the means of y across categories of x , to more complicated situations with multiple categorical and continuous or measurement x variables. t tests for hypotheses regarding a single mean (one-sample) or a pair of means (two-sample) correspond to elementary forms of ANOVA.

Rank-based nonparametric tests, including sign, Mann–Whitney and Kruskal–Wallis, take a different approach to comparing distributions. These tests make weaker assumptions about measurement, distribution shape and spread. Consequently, they remain valid under a wider range of conditions than ANOVA and its parametric relatives. Careful analysts sometimes use parametric and nonparametric tests together, checking to see whether both point toward similar conclusions. Further troubleshooting is called for when parametric and nonparametric results disagree.

anova is one of Stata’s model-fitting commands. Like the others, it has considerable flexibility encompassing a wide variety of models. **anova** can fit one-way and N -way ANOVA or analysis of covariance (ANCOVA) for balanced and unbalanced designs, including designs with missing cells. It can also fit factorial, nested, mixed or repeated-measures designs. One follow-up command, **predict**, calculates predicted values, several types of residuals, and assorted standard errors and diagnostic statistics after **anova**. Another followup command, **test**, obtains tests of user-specified null hypotheses. Both **test** and **predict** work in similar fashion with other Stata model-fitting commands, such as **regress** (Chapter 7).

The following menu choices give access to most operations described in this chapter:

Statistics > Summaries, tables, & tests > Classical tests of hypotheses

Statistics > Summaries, tables, & tests > Nonparametric tests of hypotheses

Statistics > Linear models and related > ANOVA / MANOVA

Statistics > Postestimation > Predictions, residuals, etc.

Graphics > Twoway graph (scatter, line, etc.)

Example Commands

- **anova y x1 x2**
Performs two-way ANOVA, testing for differences among the means of *y* across categories of *x1* and *x2*.
- **anova y x1 x2 x1#x2**
Performs a two-way factorial ANOVA, including both the main and interaction (*x1#x2*) effects of categorical variables *x1* and *x2*. Exactly the same model could be specified using factorial notation by the command **anova y x1##x2**, where “##” invokes not only the *x1#x2* interaction but also any lower-level interaction and main effects involving these variables (in this simple example, only the main effects of *x1* and *x2*).
- **anova y x1##x2##x3**
Performs a three-way factorial ANOVA, including the three-way interaction *x1#x2#x3*, as well as all two-way interactions (*x1#x2*, *x1#x3* and *x2#x3*) and main effects(*x1*, *x2* and *x3*).
- **anova reading curriculum / teacher|curriculum /**
Fits a nested model to test the effects of three types of curriculum on students’ reading ability (*reading*). *teacher* is nested within *curriculum* (*teacher|curriculum*) because several different teachers were assigned to each curriculum. The *Base Reference Manual* provides other nested ANOVA examples, including a split-plot design.
- **anova headache subject medication, repeated(medication)**
Fits a repeated-measures ANOVA model to test the effects of three types of headache medication (*medication*) on the severity of subjects’ headaches (*headache*). The sample consists of 20 subjects who report suffering from frequent headaches. Each subject tried each of the three medications at separate times during the study.
- **anova y x1 x2 c.x3 c.x4 x2#c.x3**
- **regress**
Performs analysis of covariance (ANCOVA) with four independent variables, two of them (*x1* and *x2*) categorical and two of them (*x3* and *x4*) continuous. Includes the *x2#c.x3* interaction. The follow-up **regress** command, with no arguments, replays results in regression-table form.
- **kwallis y, by(x)**
Performs a Kruskal–Wallis test of the null hypothesis that *y* has identical rank distributions across the *k* categories of *x* (*k* > 2).
- **oneway y x**
Performs a one-way analysis of variance (ANOVA), testing for differences among the means of *y* across categories of *x*. The same analysis, with a different output table, is produced by **anova y x**.
- **oneway y x, tabulate scheffe**
Performs one-way ANOVA, including a table of sample means and Scheffé multiple-comparison tests in the output.

. ranksum *y*, by(*x*)

Performs a Wilcoxon rank-sum test (also known as a Mann–Whitney *U* test) of the null hypothesis that *y* has identical rank distributions for both categories of dichotomous variable *x*. If we assume that both rank distributions possess the same shape, this amounts to a test for whether the two medians of *y* are equal.

. serrbar *ymean se x, scale(2)*

Constructs a standard-error-bar chart from a dataset of means. Variable *ymean* holds the group means of *y*; *se* the standard errors; and *x* the values of categorical variable *x*. **scale(2)** asks for bars extending to ± 2 standard errors around each mean (default is ± 1).

. signrank *y1 = y2*

Performs a Wilcoxon matched-pairs signed-rank test for the equality of the rank distributions of *y1* and *y2*. We could test whether the median of *y1* differs from a constant such as 23.4 by typing the command **signrank y1 = 23.4**.

. signtest *y1 = y2*

Tests the equality of the medians of *y1* and *y2* (assuming matched data; that is, both variables measured on the same sample of observations). Typing **signtest y1 = 5** would perform a sign test of the null hypothesis that the median of *y1* equals 5.

. ttest *y = 5*

Performs a one-sample *t* test of the null hypothesis that the population mean of *y* equals 5.

. ttest *y1 = y2*

Performs a one-sample (paired difference) *t* test of the null hypothesis that the population mean of *y1* equals that of *y2*. The default form of this command assumes that the data are paired. With unpaired data (*y1* and *y2* are measured from two independent samples), add the option **unpaired**.

. ttest *y, by(x) unequal*

Performs a two-sample *t* test of the null hypothesis that the population mean of *y* is the same for both categories of variable *x*. Does not assume that the populations have equal variances. (Without the **unequal** option, **ttest** does assume equal variances.)

One-Sample Tests

One-sample *t* tests have two seemingly different applications:

1. Testing whether a sample mean \bar{y} differs significantly from an hypothesized value μ_0 .
2. Testing whether the means of y_1 and y_2 , two variables measured over the same set of observations, differ significantly from each other. This is equivalent to testing whether the mean of a difference score variable created by subtracting y_1 from y_2 equals zero.

We use essentially the same formulas for either application, although the second starts with information on two variables instead of one.

The data in *writing.dta* were collected to evaluate a college writing course based on word processing (Nash and Schwartz 1987). Measures such as the number of sentences completed in timed writing were collected both before and after students took the course. The researchers wanted to know whether the post-course measures showed improvement.

Contains data from C:\data\writing.dta				Nash and Schwartz (1987)
obs:	24 <th>vars:</th> <td>9</td> <th>15 Mar 2012 09:17</th>	vars:	9	15 Mar 2012 09:17
size:	216 <th data-cs="3" data-kind="parent"></th> <th data-kind="ghost"></th> <th data-kind="ghost"></th>			
variable	storage type	display format	value label	variable label
id	byte	%8.0g	s1b1	Student ID
preS	byte	%8.0g		# of sentences (pre-test)
preP	byte	%8.0g		# of paragraphs (pre-test)
preC	byte	%8.0g		Coherence scale 0-2 (pre-test)
preE	byte	%8.0g		Evidence scale 0-6 (pre-test)
postS	byte	%8.0g		# of sentences (post-test)
postP	byte	%8.0g		# of paragraphs (post-test)
postC	byte	%8.0g		Coherence scale 0-2 (post-test)
postE	byte	%8.0g		Evidence scale 0-6 (post-test)

Sorted by:

Suppose that we knew that students in previous years were able to complete an average of 10 sentences. Before examining whether the students in *writing.dta* improved during the course, we might want to learn whether at the start of the course they were essentially like earlier students — in other words, whether their pre-test (*preS*) mean differs significantly from the mean of previous students (10). To see a one-sample *t* test of $H_0: \mu = 10$, type

One-sample t test						
Variable	Obs	Mean	Std. Err.	Std. Dev.	[95% Conf. Interval]	
preS	24	10.79167	.9402034	4.606037	8.846708	12.73663
mean = mean(preS)					t =	0.8420
Ho: mean = 10					degrees of freedom =	23
Ha: mean < 10						
Pr(T < t) = 0.7958						
Ha: mean != 10						
Pr(T > t) = 0.4084						
Ha: mean > 10						
Pr(T > t) = 0.2042						

The notation $\text{Pr}(T < t)$ means “probability of a *t*-distribution value less than the observed *t*, if H_0 were true”— that is, the one-tail test probability. The two-tail probability of a greater absolute *t* appears as $\text{Pr}(|T| > |t|) = 0.4084$. Because this probability is high, we have no reason to reject $H_0: \mu = 10$. Note that **ttest** automatically provides a 95% confidence interval for the mean, and this confidence interval includes the null-hypothesis value 10. We could see a different confidence interval, such as 90%, by adding a **level(90)** option to this command.

A nonparametric counterpart, the sign test, employs the binomial distribution to test hypotheses about single medians. For example, we could test whether the median of *preS* equals 10. **signtest** gives us no reason to reject that null hypothesis either.

```
. signtest pres = 10
Sign test

      sign | observed   expected
    positive |       12       11
    negative |       10       11
      zero   |       2        2
    all     |      24      24

One-sided tests:
  Ho: median of pres - 10 = 0 vs.
  Ha: median of pres - 10 > 0
  Pr(#positive >= 12) =
    Binomial(n = 22, x >= 12, p = 0.5) = 0.4159

  Ho: median of pres - 10 = 0 vs.
  Ha: median of pres - 10 < 0
  Pr(#negative >= 10) =
    Binomial(n = 22, x >= 10, p = 0.5) = 0.7383

Two-sided test:
  Ho: median of pres - 10 = 0 vs.
  Ha: median of pres - 10 != 0
  Pr(#positive >= 12 or #negative >= 12) =
    min(1, 2*Binomial(n = 22, x >= 12, p = 0.5)) = 0.8318
```

Like **ttest**, **signtest** includes right-tail, left-tail, and two-tail probabilities. Unlike the symmetrical *t* distributions used by **ttest**, however, the binomial distributions used by **signtest** have different left- and right-tail probabilities. In this example, only the two-tail probability matters because we were testing whether the *writing.dta* students “differ” from the null-hypothesis median of 10.

Next, we can test for improvement during the course by testing the null hypothesis that the mean number of sentences completed before and after the course (that is, the means of *preS* and *postS*) are equal. The **ttest** command accomplishes this as well, finding a significant improvement.

```
. ttest postS = pres
Paired t test

      Variable |   obs      Mean   Std. Err.   Std. Dev. [95% Conf. Interval]
      posts   |   24    26.375  1.693779  8.297787  22.87115  29.87885
      pres    |   24    10.79167 .9402034  4.606037  8.846708  12.73663
      diff    |   24    15.58333  1.383019  6.775382  12.72234  18.44433

      mean(diff) = mean(posts - pres)          t = 11.2676
      Ho: mean(diff) = 0           degrees of freedom = 23
      Ha: mean(diff) < 0          Ha: mean(diff) != 0          Ha: mean(diff) > 0
      Pr(T < t) = 1.0000          Pr(|T| > |t|) = 0.0000          Pr(T > t) = 0.0000
```

Because we expect “improvement,” not just “difference” between the *preS* and *postS* means, a one-tail test is appropriate. The displayed right-tail probability rounds off to zero. Students’ mean sentence completion does significantly improve. Based on this sample, we are 95% confident that it improves by between 12.7 and 18.4 sentences.

t tests ordinarily assume that variables are normally distributed around their group means. This assumption usually is not critical because the tests are moderately robust. When nonnormality involves severe outliers, however, or occurs in small samples, we might be safer turning to medians instead of means and employing a nonparametric test that does not assume normality. The Wilcoxon signed-rank test, for example, assumes only that the distributions are symmetrical and continuous. Applying a signed-rank test to these data yields essentially the same conclusion as **ttest**: that students' sentence completion significantly improved. Because both tests agree on this conclusion, we can state it with more assurance.

```
. signrank posts = pres
Wilcoxon signed-rank test

sign |   obs   sum ranks   expected
-----+
positive |    24      300      150
negative |     0       0      150
zero     |     0       0       0
-----+
all      |    24      300      300

unadjusted variance      1225.00
adjustment for ties      -1.63
adjustment for zeros      0.00
-----+
adjusted variance        1223.38

Ho: posts = pres
      z =  4.289
  Prob > |z| =  0.0000
```

Two-Sample Tests

The remainder of this chapter draws examples from a survey of college undergraduates by Ward and Ault (1990).

```
. use C:\data\student2.dta
. describe

Contains data from C:\data\student2.dta
obs:           243                               Student survey (Ward 1990)
vars:            18                               16 Mar 2012 13:25
size:          5,346

variable   storage   display   value   variable
name      type      format    label    label
id         int       %8.0g
year       byte      %9.0g    year
age        byte      %8.0g
gender     byte      %9.0g    s
relig      byte      %8.0g    v4
drink      byte      %9.0g
gpa        float     %9.0g
grades     byte      %8.0g    grades
greek      byte      %9.0g    greek
live       byte      %8.0g    v10
miles      byte      %8.0g
study      byte      %8.0g
athlete    byte      %9.0g    athlete
```

employed	byte	%8.0g	employ	Are you employed?
allnight	byte	%8.0g	allnight	How often study all night?
ditch	byte	%8.0g	times	How many class/month ditched?
hsdrink	byte	%9.0g		High school drinking scale
aggress	byte	%9.0g		Aggressive behavior scale

Sorted by: year

About 19% of these students belong to a fraternity or sorority. On campus, these organizations and their members are commonly referred to as “Greeks” not with any reference to nationality, but because most of the organizations have names composed of Greek letters.

. tabulate greek

Belong to fraternity or sorority	Freq.	Percent	Cum.
non-Greek	196	80.66	80.66
Greek	47	19.34	100.00
Total	243	100.00	

Another variable, *drink*, measures how often and heavily a student drinks alcohol, on a 33-point scale. Campus rumors might lead one to suspect that fraternity/sorority members tend to differ from other students in their drinking behavior. Box plots comparing the median *drink* values of members and nonmembers, and a bar chart comparing their means, both appear consistent with such rumors. Figure 6.1 combines these two separate plot types in one image, after using `ylabel(0(5)25)` to set a common y-axis scale for comparability.

```
. graph box drink, over(greek) ylabel(0(5)35) saving(fig06_01a)
. graph bar (mean) drink, over(greek) ylabel(0(5)35)
    saving(fig06_01b)
. graph combine fig06_01a.gph fig06_01b.gph, col(2) iscale(1.05)
```

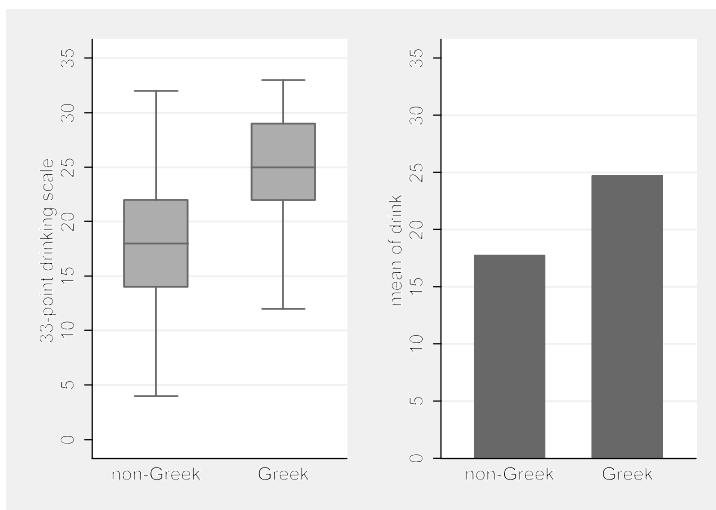


Figure 6.1

The **ttest** command, applied earlier to one-sample and paired-difference tests, can perform two-sample tests as well. In this application its general syntax is **ttest measurement, by(categorical)**. For example,

```
. ttest drink, by(greek)
```

Two-sample t test with equal variances

Group	Obs	Mean	Std. Err.	Std. Dev.	[95% Conf. Interval]
non-Gree Greek	196 47	17.7602 24.7234	.4575013 .7124518	6.405018 4.884323	16.85792 23.28931 18.66249 26.1575
combined	243	19.107	.431224	6.722117	18.25756 19.95643
diff		-6.9632	.9978608		-8.928842 -4.997558
diff = mean(non-Gree) - mean(Greek)					t = -6.9781
Ho: diff = 0					degrees of freedom = 241
Ha: diff < 0		Ha: diff != 0		Ha: diff > 0	
Pr(T < t) = 0.0000		Pr(T > t) = 0.0000		Pr(T > t) = 1.0000	

As the output notes, this *t* test rests on an equal-variance assumption. But the fraternity and sorority members' sample standard deviation appears somewhat lower — they are more alike than nonmembers in their reported drinking behavior. To perform a similar test without assuming equal variances, add the option **unequal**:

```
. ttest drink, by(greek) unequal
```

Two-sample t test with unequal variances

Group	Obs	Mean	Std. Err.	Std. Dev.	[95% Conf. Interval]
non-Gree Greek	196 47	17.7602 24.7234	.4575013 .7124518	6.405018 4.884323	16.85792 23.28931 18.66249 26.1575
combined	243	19.107	.431224	6.722117	18.25756 19.95643
diff		-6.9632	.8466965		-8.645773 -5.280627
diff = mean(non-Gree) - mean(Greek)					t = -8.2240
Ho: diff = 0					Satterthwaite's degrees of freedom = 88.22
Ha: diff < 0		Ha: diff != 0		Ha: diff > 0	
Pr(T < t) = 0.0000		Pr(T > t) = 0.0000		Pr(T > t) = 1.0000	

Adjusting for unequal variances does not alter our basic conclusion that Greeks (fraternity/sorority members) and non-Greeks are significantly different. We can further check this conclusion by trying a nonparametric Mann–Whitney *U* test, also known as a Wilcoxon rank-sum test. Assuming that the rank distributions have similar shape, the rank-sum test indicates that we can reject the null hypothesis of equal population medians.

```
. ranksum drink, by(greek)
```

Two-sample Wilcoxon rank-sum (Mann-Whitney) test

greek	obs	rank sum	expected
non-Greek	196	21111	23912
Greek	47	8535	5734
combined	243	29646	29646
unadjusted variance	187310.67		
adjustment for ties	-472.30		
adjusted variance	186838.36		

Ho: drink(greek==non-Greek) = drink(greek==Greek)
z = -6.480
Prob > |z| = 0.0000

One-Way Analysis of Variance (ANOVA)

Analysis of variance (ANOVA) provides another way, more general than *t* tests, to test for differences among means. The simplest case, one-way ANOVA, tests whether the means of *y* differ across categories of *x*. One-way ANOVA can be performed by a **oneway** command with the general form **oneway measurement categorical**. For example,

```
. oneway drink greek, tabulate
```

Belong to fraternity or sorority	Summary of 33-point drinking scale		
	Mean	Std. Dev.	Freq.
non-Greek	17.760204	6.4050179	196
Greek	24.723404	4.8843233	47
Total	19.106996	6.7221166	243

Source	Analysis of Variance			F	Prob > F
	SS	df	MS		
Between groups	1838.08426	1	1838.08426	48.69	0.0000
Within groups	9097.13385	241	37.7474433		
Total	10935.2181	242	45.1868517		

Bartlett's test for equal variances: chi2(1) = 4.8378 Prob>chi2 = 0.028

The **tabulate** option produces a table of means and standard deviations in addition to the analysis of variance table itself. One-way ANOVA with a dichotomous *x* variable is equivalent to a two-sample *t* test, and its *F* statistic equals the corresponding *t* statistic squared. **oneway** offers more options and processes faster, but it lacks an **unequal** option for relaxing the equal-variances assumption.

oneway does, however, formally test the equal-variances assumption using Bartlett's χ^2 . A low Bartlett's probability implies that an equal-variance assumption is implausible, in which case we should not trust the ANOVA *F* test results. In the **oneway drink belong** example above, Bartlett's *p* = .028 casts doubt on the ANOVA's validity.

One-way ANOVA's real value lies not in two-sample comparisons, but in comparisons of three or more means. For example, we could test whether mean drinking behavior varies by year in college. The term "freshman" refers to first-year students, not necessarily male.

```
. oneway drink year, tabulate scheffe
```

Year in college	Summary of 33-point drinking scale		
	Mean	Std. Dev.	Freq.
Freshman	18.975	6.9226033	40
Sophomore	21.169231	6.5444853	65
Junior	19.453333	6.2866081	75
Senior	16.650794	6.6409257	63
Total	19.106996	6.7221166	243

Source	Analysis of Variance			F	Prob > F
	SS	df	MS		
Between groups	666.200518	3	222.066839	5.17	0.0018
Within groups	10269.0176	239	42.9666008		
Total	10935.2181	242	45.1868517		

Bartlett's test for equal variances: $\text{chi2}(3) = 0.5103 \text{ Prob>chi2} = 0.917$

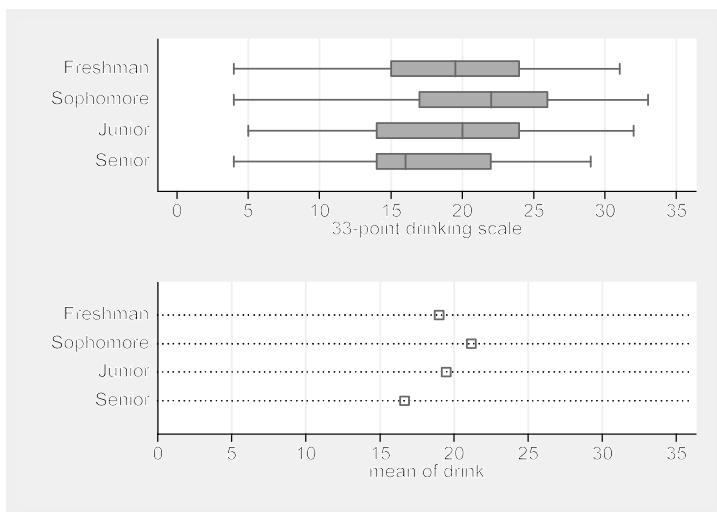
Comparison of 33-point drinking scale by Year in college (Scheffe)

Row Mean - Col Mean	Freshman	Sophomor	Junior
Sophomor	2.19423 0.429		
Junior	.478333 0.987	-1.7159 0.498	
Senior	-2.32421 0.382	-4.51844 0.002	-2.80254 0.103

We can reject the hypothesis of equal means ($p = .0018$), but not the hypothesis of equal variances ($p = .917$). The latter is good news regarding the ANOVA's validity.

The horizontal box plots (**graph hbox**) in Figure 6.2 support this conclusion, showing similar variation within each category. For this image the box plots are combined with a dot chart (**graph dot (mean)**) depicting means within each category. The combined image shows that differences among medians (top) and differences among means (bottom) follow similar patterns. Dot charts serve much the same purpose as bar charts: visually comparing statistical summaries of one or more measurement variables. The organization and options for dot charts and bar charts are broadly similar, including the choices of statistical summaries. Type **help graph dot** for details.

```
. graph hbox drink, over(year) ylabel(0(5)35) saving(fig06_02a)
. graph dot (mean) drink, over(year) ylabel(0(5)35, grid)
    marker(1, msymbol(sh)) saving(fig06_02b)
. graph combine fig06_02a.gph fig06_02b.gph, row(2) iscale(1.05)
```

**Figure 6.2**

The **scheffe** option (Scheffé multiple-comparison test) with our **oneway** command produced a table showing the differences between each pair of means. The freshman mean equals 18.975 and the sophomore mean equals 21.16923, so the sophomore–freshman difference is $21.16923 - 18.975 = 2.19423$, not statistically distinguishable from zero ($p = .429$). Of the six contrasts in this table, only the senior–sophomore difference, $16.6508 - 21.1692 = -4.5184$, is significant ($p = .002$). Thus, our overall conclusion that these four groups' means are not the same arises mainly from the contrast between seniors (the lightest drinkers) and sophomores (the heaviest).

oneway offers three multiple-comparison options: **scheffe**, **bonferroni** and **sidak** (see *Base Reference Manual* for definitions). The Scheffé test remains valid under a wider variety of conditions, although it is sometimes less sensitive.

The Kruskal–Wallis test (**kwallis**), a K -sample generalization of the two-sample rank-sum test, provides a nonparametric alternative to one-way ANOVA. It tests the null hypothesis of equal population medians.

```
. kwallis drink, by(year)
```

Kruskal-Wallis Equality-of-Populations Rank Test

year	Obs	Rank Sum
Freshman	40	4914.00
Sophomore	65	9341.50
Junior	75	9300.50
Senior	63	6090.00

chi-squared = 14.453 with 3 d.f.
 probability = 0.0023

chi-squared with ties = 14.490 with 3 d.f.
 probability = 0.0023

These **kwallis** results ($p = .0023$) agree with our **oneway** findings of significant differences in *drink* by year in college. Kruskal-Wallis is generally safer than ANOVA if we have reason to doubt ANOVA's equal-variances or normality assumptions, or if we suspect problems caused by outliers. **kwallis**, like **ranksum**, makes the weaker assumption of similar-shaped rank distributions within each group. In principle, **ranksum** and **kwallis** should produce similar results when applied to two-sample comparisons, but in practice this is true only if the data contain no ties. **ranksum** incorporates an exact method for dealing with ties, which makes it preferable for two-sample problems.

Two- and N-Way Analysis of Variance

One-way ANOVA examines how the means of measurement variable *y* vary across categories of one other variable *x*. *N*-way ANOVA generalizes this approach to deal with two or more categorical *x* variables. For example, we might consider how drinking behavior varies not only by fraternity or sorority membership, but also by gender. We start by examining a two-way table of means:

```
. table greek gender, contents(mean drink) row col
```

Belong to fraternity or sorority	Gender (male)		
	Female	Male	Total
non-Greek	16.51724	19.5625	17.7602
Greek	22.44444	26.13793	24.7234
Total	17.31343	21.31193	19.107

It appears that in this sample, males drink more than females and fraternity/sorority members drink more than nonmembers. The Greek/non-Greek difference appears similar among males and females. **anova**, Stata's *N*-way ANOVA command, can test for significant differences among these means attributable to belonging to a fraternity or sorority, gender, or the interaction of Greek membership and gender (written *greek#gender*).

```
. anova drink greek gender greek#gender
```

	Number of obs =	243	R-squared =	0.2221
	Root MSE	= 5.96592	Adj R-squared =	0.2123
Source	Partial SS	df	MS	F
Model	2428.67237	3	809.557456	22.75
greek	1406.2366	1	1406.2366	39.51
gender	408.520097	1	408.520097	11.48
greek#gender	3.78016612	1	3.78016612	0.11
Residual	8506.54574	239	35.5922416	0.7448
Total	10935.2181	242	45.1868517	

In this example of two-way factorial ANOVA, the output shows significant main effects for *greek* ($p = .0000$) and *gender* ($p = .0008$), but their interaction contributes little to the model ($p = .7448$). Because this interaction cannot be distinguished from zero, we might prefer to fit a simpler model without the interaction term.

To include any interaction term with **anova**, specify the variable names joined by # (or ## for a *factorial interaction*). Unless the number of observations with each combination of *x* values is the same (a condition called balanced data), it can be hard to interpret the main effects in a model that also includes interactions. This does not mean that the main effects in such models are unimportant, however. Regression analysis might help to make sense of complicated ANOVA results, as seen in the following section.

Factor Variables and Analysis of Covariance (ANCOVA)

anova and many other Stata estimation commands allow independent variables specified in *factor variable* notation. The prefix **i.** written before the name of an independent variable tells Stata to include indicator (binary) variables for levels of a categorical variable, as if each category comprised its own dichotomous predictor. Categorical variables marked by the **i.** prefix must have non-negative integer values, from 0 to 32,740. **anova** by default views all independent variables as categorical, so typing

```
. anova drink greek year greek#year
```

invokes the same model as typing

```
. anova drink i.greek i.year i.greek##i.year
```

	Number of obs =	243	R-squared =	0.2265
	Root MSE	= 5.99962	Adj R-squared =	0.2034
Source	Partial SS	df	MS	F
Model	2476.29537	7	353.756482	9.83
greek	1457.93596	1	1457.93596	40.50
year	217.492051	3	72.4973502	2.01
greek#year	148.508479	3	49.5028264	1.38
Residual	8458.92273	235	35.9954159	
Total	10935.2181	242	45.1868517	

As often the case with ANOVA, we could get a more direct view of the underlying model by re-expressing the analysis as regression. Stata does this easily: simply type **regress** with no arguments immediately after **anova**.

. **regress**

Source	SS	df	MS	Number of obs =	243
Model	2476.29537	7	353.756482	F(7, 235) =	9.83
Residual	8458.92273	235	35.9954159	Prob > F =	0.0000
Total	10935.2181	242	45.1868517	R-squared =	0.2265
				Adj R-squared =	0.2034
				Root MSE	= 5.9996
drink	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
1.greek	7.805556	3.162076	2.47	0.014	1.575917 14.03519
year					
2	1.138889	1.322791	0.86	0.390	-1.467156 3.744934
3	.3648776	1.268844	0.29	0.774	-2.134884 2.864639
4	-3.043501	1.295774	-2.35	0.020	-5.596319 -.4906827
greek#year					
1 2	-.7859477	3.586922	-0.22	0.827	-7.852579 6.280683
1 3	-3.614878	3.58588	-1.01	0.314	-10.67945 3.4497
1 4	1.643501	3.778548	0.43	0.664	-5.800655 9.087657
_cons	18.19444	.9999363	18.20	0.000	16.22446 20.16443

Note that sums of squares, F test, R^2 and other details are identical for these equivalent **anova** and **regress** analyses. The **regress** table gives more detail, however. We see that each value of *year* is treated as its own predictor. First-year college students form the omitted category in this table, so the coefficients on years 2, 3 and 4 express contrasts with these first-year students. Second-year non-Greek students drink somewhat more (+1.14), whereas fourth-year non-Greek students drink considerably less (-3.04) compared with first-year students. These *year* coefficients correspond to coefficients on dummy variables coded 1 for that particular year and 0 for any other. The *greek* coefficient corresponds to the coefficient on a dummy variable coded 1 for Greek and 0 for non-Greek students.

Analysis of covariance (ANCOVA) extends N -way ANOVA to encompass a mix of categorical and continuous x variables. The **c.** prefix identifies an independent variable as continuous, in which case its values are treated as measurements rather than separate, unordered categories. We might have treated *year* as continuous:

```
. anova drink i.greek c.year i.greek#c.year
```

		Number of obs =	243	R-squared =	0.1965
		Root MSE	= 6.06334	Adj R-squared =	0.1864
Source	Partial SS	df	MS	F	Prob > F
Model	2148.60352	3	716.201174	19.48	0.0000
greek	186.474269	1	186.474269	5.07	0.0252
year	147.628787	1	147.628787	4.02	0.0462
greek#year	.203073456	1	.203073456	0.01	0.9408
Residual	8786.61458	239	36.7640778		
Total	10935.2181	242	45.1868517		

```
. regress
```

Source	SS	df	MS	Number of obs =	243
Model	2148.60352	3	716.201174	F(3, 239) =	19.48
Residual	8786.61458	239	36.7640778	Prob > F =	0.0000
Total	10935.2181	242	45.1868517	R-squared =	0.1965
				Adj R-squared =	0.1864
				Root MSE =	6.0633

drink	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
1.greek	6.776657	3.00897	2.25	0.025	.8491681 12.70415
year	-1.103421	.4068558	-2.71	0.007	-1.904902 -.3019392
greek#c.year					
1	.0789217	1.061895	0.07	0.941	-2.012947 2.17079
_cons	20.69328	1.164985	17.76	0.000	18.39833 22.98823

This version with *year* as continuous variable (*c.year*) instead of a categorical variable (*i.year*) results in a simpler model with more degrees of freedom, but the adjusted R^2 shows the continuous version does not fit as well (.1864 versus .2034). The categorical version found that drinking is higher (+1.14) in year 2 compared with year 1, slightly higher (+.36) in year 3 compared with year 1, but much lower (-.04) in year 4 compared with year 1. The continuous version just smoothed the up-and-down details into an average decrease of -1.10 per year.

Treating *year* as a continuous or categorical variable can be the analyst's call, based on substantive or statistical reasoning. Other variables such as student grade point average (*gpa*) are unambiguously continuous, however. When we include *gpa* among the independent variables, we find that it, too, is related to drinking behavior. This model omits the interaction effects, which have not proven to be significant. Because categorical variables are the default for **anova**, the *i.* prefixes can be omitted with *greek* and *gender*.

```
. anova drink greek gender c.gpa
```

		Number of obs =	218	R-squared =	0.2970
		Root MSE	= 5.68939	Adj R-squared =	0.2872
Source	Partial SS	df	MS	F	Prob > F
Model	2927.03087	3	975.676958	30.14	0.0000
greek	1489.31999	1	1489.31999	46.01	0.0000
gender	405.137843	1	405.137843	12.52	0.0005
gpa	407.0089	1	407.0089	12.57	0.0005
Residual	6926.99206	214	32.3691218		
Total	9854.02294	217	45.4102439		

From this analysis we know that a significant relationship exists between *drink* and *gpa* when we control for *greek* and *gender*. Beyond their *F* tests for statistical significance, however, ANOVA or ANCOVA tables do not provide much descriptive information about how variables are related. Regression does a better job at this.

```
. regress
```

Source	SS	df	MS	Number of obs =	218
Model	2927.03087	3	975.676958	<i>F</i> (3, 214) =	30.14
Residual	6926.99206	214	32.3691218	Prob > F =	0.0000
Total	9854.02294	217	45.4102439	R-squared =	0.2970
				Adj R-squared =	0.2872
				Root MSE =	5.6894

drink	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
1.greek	6.547869	.9653204	6.78	0.000	4.645116 8.450623
1.gender	2.808418	.7938269	3.54	0.000	1.243697 4.373139
gpa	-3.038966	.8570168	-3.55	0.000	-4.728241 -1.34969
_cons	24.72871	2.539529	9.74	0.000	19.72301 29.7344

Predicted Values and Error-Bar Charts

After **anova**, the followup command **predict** calculates predicted values, residuals or standard errors and diagnostic statistics. One use for such statistics is in drawing graphical representations of the model results, such as an error-bar chart. For a simple illustration, we return to the one-way ANOVA of *drink* by *year*:

```
. anova drink year
```

		Number of obs =	243	R-squared =	0.0609
		Root MSE	= 6.55489	Adj R-squared =	0.0491
Source	Partial SS	df	MS	F	Prob > F
Model	666.200518	3	222.066839	5.17	0.0018
year	666.200518	3	222.066839	5.17	0.0018
Residual	10269.0176	239	42.9666008		
Total	10935.2181	242	45.1868517		

To calculate predicted means from the recent **anova**, type a command with form **predict newvar1**, where “newvar1” can be any name you want to give these means. **predict newvar2, stdp** creates a second new variable containing standard errors of the predicted means.

```
. predict drinkmean
. predict SEdrink, stdp
```

Using new variables *drinkmean* and *SEdrink*, we calculate approximate 95% confidence intervals as the means plus or minus 2 standard errors. The error-bar chart in Figure 6.3 consists of a range plot with capped spikes (**rcap**) for the error bars, overlaid by a connected-line plot (**connect**) for the means.

```
. gen drinkhi = drinkmean + 2 * SEdrink
. gen drinklo = drinkmean - 2 * SEdrink
. graph twoway rcap drinkhi drinklo year, color(maroon)
    || connect drinkmean year, lwidth(medthick) color(maroon)
    || , ylabel(14(1)23, grid gmin gmax) ytitle("Drinking scale")
    legend(off) text(18 2 "Mean `=char(177)'2SE", box margin(small))
```

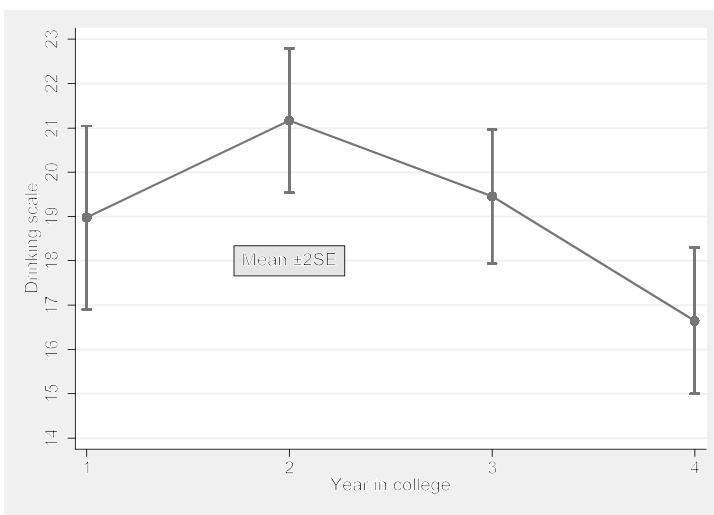


Figure 6.3

Figure 6.3 includes several other options to improve the image. Both the **rcap** and **connect** plots are given the same **color(maroon)**. Stata’s default legend is suppressed (**legend(off)**) in favor of a simple **text** box to explain that the graph shows “Mean \pm 2SE”. The plus or minus symbol \pm is ASCII character 177, represented by ‘=char(177)’ in the command. Figure 3.16 in Chapter 3 displays the complete set of ASCII characters that are available for use in Stata graphs.

Drawing Figure 6.3 in this fashion provided an introduction to the **predict** command, which has broad applications in statistical modeling. An alternative way to draw error-bar charts makes use of two other commands, **margins** and **marginsplot**. The **margins** command calculates marginal means or predictive margins after a modeling command. **marginsplot** displays these

graphically. In the example below, **margins year** obtains mean values of *drink* for each *year*. Then **marginsplot** simply graphs these means with their confidence intervals. Figure 6.4 is a plain version, but we could apply many of the standard **twoway** options to **marginsplot** if desired.

```
. margins year
. marginsplot
```

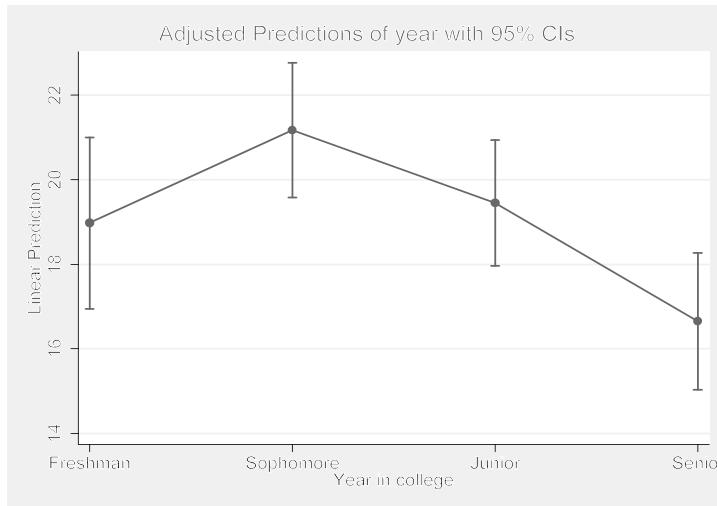


Figure 6.4

For a two-way factorial ANOVA, error-bar charts help us to visualize main and interaction effects. For this example we use the aggressive-behavior scale *aggress* as our dependent variable, in a factorial ANOVA with *gender*, *year* and the interaction term *gender#year* as predictors. In light of the nonlinear *year* effects seen in Figure 6.3/6.4, for this analysis we accept the default treatment of *year* as a categorical rather than a continuous variable. *F* tests show that *gender*, *year* and *gender#year* all have significant effects.

```
. anova aggress gender year gender#year
```

		Number of obs = 243	R-squared = 0.2503		
		Root MSE = 1.45652	Adj R-squared = 0.2280		
Source	Partial SS	df	MS	F	Prob > F
Model	166.482503	7	23.7832147	11.21	0.0000
gender	94.3505972	1	94.3505972	44.47	0.0000
year	19.0404045	3	6.34680149	2.99	0.0317
gender#year	24.1029759	3	8.03432529	3.79	0.0111
Residual	498.538073	235	2.12143861		
Total	665.020576	242	2.74801891		

We use **predict** to calculate a new variable holding the predicted means, and **predict, stdp** to calculate standard errors. Approximate high and low confidence limits equal the means plus or

minus two standard errors. To represent the *gender#year* interaction, the **graph** command for Figure 6.5 employs a **by(gender)** option that draws separate plots for males and females. Some other options illustrate control of secondary details such as marker symbols (large Diamonds), suppressed **legend** and suppressed **note**. We also draw small-margin boxes with white background around the “Mean ±2SE” text, which is carefully placed so it does not interfere with the data in either sub-plot,

```
. predict aggmean
. predict SEagg, stdp
. gen agghi = aggmean + 2 * SEagg
. gen agglo = aggmean - 2 * SEagg
. graph twoway rcap agghi agglo year
    || connect aggmean year, lwidth(medthick) msymbol(D)
    || , by(gender, legend(off) note(""))
    ytitle("Aggressive behavior scale")
    text(1.7 2 "Mean `=char(177)'2SE", box
        margin(small) bfcolor(white))
```

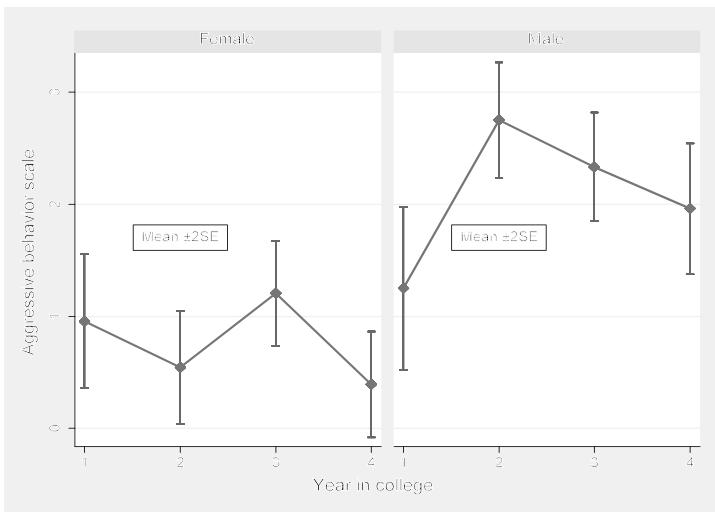
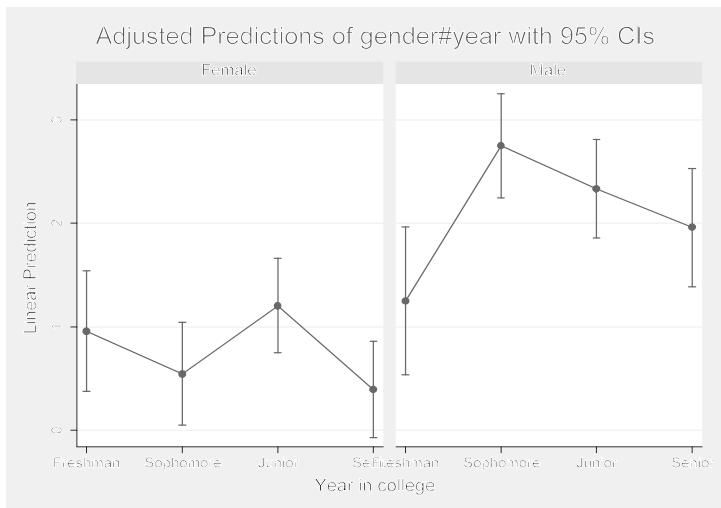


Figure 6.5

A basic but similar error-bar chart could have been created quickly using **margins** and **marginsplot**. The command **margins gender#year** calculates predicted values or means of *drink* for every combination of *gender* and *year*. **marginsplot, by(gender)** then plots these means with their confidence intervals separately by gender (Figure 6.6).

```
. margins gender#year
. marginsplot, by(gender)
```

**Figure 6.6**

Substantively, Figure 6.5 or 6.6 add details about the gender and interaction effects seen by **anova**. Female means on the aggressive-behavior scale fluctuate at comparatively low levels during the four years of college. Male means are higher throughout, with a second-year peak that resembles the pattern seen earlier for drinking (Figures 6.2 and 6.3). Thus, the relationship between *aggress* and *year* is different for males and females. Error-bar charts visually complement the **anova** or **regress** tables they are based on. While the tables confirm which effects are significant and give numerical details, the charts help to picture what these effects mean.

Linear Regression Analysis

Stata offers an exceptionally broad range of regression methods. A partial list of the possibilities can be seen by typing **help regress**. This chapter focuses on simple and multiple regression using the method of ordinary least squares (OLS), accomplished by **regress** and related commands. Graphical and post-regression diagnostic methods extend the basic regression toolkit to help interpret results, or to detect and cope with analytical complications. **regress** can also accomplish some non-OLS analyses including weighted least squares. Other non-OLS regression methods appear in Chapter 8 and the remainder of this book.

The following menus access most of the operations discussed:

Statistics > Linear models and related > Linear regression

Statistics > Linear models and related > Regression diagnostics

Graphics > Twoway graph (scatter, line, etc.)

Statistics > Postestimation > Predictions, residuals, etc.

Statistics > Postestimation > Marginal means and predictive margins

Statistics > Postestimation > Margins plots and profile plots

This chapter illustrates some ways to graph regression models. Many further examples are found in *Interpreting and Visualizing Regression Models Using Stata* (Mitchell 2012).

Example Commands

. regress y x

Performs ordinary least squares (OLS) regression of variable *y* on one predictor, *x*.

. regress y x if ethnic == 3 & income > 50 & income < .

Regresses *y* on *x* using only that subset of the data for which variable *ethnic* equals 3 and *income* is greater than 50 (but not missing).

. predict yhat

Generates a new variable (here arbitrarily named *yhat*) equal to the predicted values from the most recent regression.

- . **`predict e, resid`**
Generates a new variable (here arbitrarily named *e*) equal to the residuals from the most recent regression.
- . **`predict new, cooksd`**
Generates a new variable equal to Cook's distance *D*, summarizing how much each observation influences the fitted model.
- . **`predict new, covratio`**
Generates a new variable equal to Belsley, Kuh and Welsch's *COVRATIO* statistic. *COVRATIO* measures the *i*th case's influence upon the variance–covariance matrix of the estimated coefficients.
- . **`predict Dfx1, dfbeta(x1)`**
Generates *DFBETAS* case statistics measuring how much each observation affects the coefficient on predictor *x1*. To create a complete set of *DFBETAS* for all predictors in the model, simply type the command **`dfbeta`** without arguments.
- . **`predict new, dfits`**
Generates *DFITS* case statistics, summarizing the influence of each observation on the fitted model (similar in purpose to Cook's *D* and Welsch's *W*).
- . **`graph twoway lfit y x || scatter y x`**
Draws the simple regression line (**`lfit`** or linear fit) with a scatterplot of *y* vs. *x*.
- . **`graph twoway mspline yhat x || scatter y x`**
Draws a simple regression line with a scatterplot of *y* vs. *x* by connecting (with a smooth cubic spline curve) the regression's predicted values (in this example named *yhat*). There are many alternative ways to draw regression lines, including **`mspline`**, **`mband`**, **`line`**, **`lfit`**, **`lfitci`**, **`qfit`** and **`marginsplot`**. Each has its own advantages and options.
- . **`graph twoway scatter e yhat, yline(0)`**
Draws a residual versus predicted values plot using the variables *e* and *yhat*. Alternatively, typing **`rvfplot`** (residuals versus fitted) after any regression will produce such a graph.
- . **`regress y x1 x2 x3`**
Performs multiple regression of *y* on three predictor variables, *x1*, *x2* and *x3*.
- . **`test x1 x2`**
Performs an *F* test of the null hypothesis that coefficients on *x1* and *x2* both equal zero in the most recent regression model.
- . **`regress y x1 x2 x3, vce(robust)`**
Calculates robust (Huber/White) estimates of standard errors. See the *User's Guide* for details. The **`vce(robust)`** option works with many other model fitting commands as well.
- . **`regress y x1 x2 x3, beta`**
Performs multiple regression and includes standardized regression coefficients (beta weights) in the output table.

- **correlate x1 x2 x3 y**

Displays a matrix of Pearson correlations, using only observations with no missing values on all of the variables specified. Adding the option **covariance** produces a variance–covariance matrix instead of correlations.

- **pwcorr x1 x2 x3 y, sig star(.05)**

Displays a matrix of Pearson correlations, using pairwise deletion of missing values and showing probabilities from *t* tests of null hypothesis $H_0:p = 0$, for each correlation. Statistically significant correlations (in this example, $p < .05$) are indicated by stars (*).

- **graph matrix x1 x2 x3 y, half**

Draws a scatterplot matrix. Because their variable lists are the same, this example yields a scatterplot matrix having the same organization as the correlation matrix produced by the preceding **pwcorr** command. Listing the dependent (*y*) variable last creates a matrix in which the bottom row forms a series of *y*-versus-*x* plots.

- **estat hettest**

Performs Cook and Weisberg's test for heteroskedasticity. If we have reason to suspect that heteroskedasticity is a function of a particular predictor *x1*, we could focus on that predictor by typing **estat hettest x1**. Type **help regress postestimation** for a complete list of options available after **regress**. Different postestimation choices exist for other modeling methods.

- **estat ovtest, rhs**

Performs the Ramsey regression specification error test (*RESET*) for omitted variables. The option **rhs** calls for using powers of the right-hand-side variables, instead of powers of predicted *y* (default).

- **estat vif**

Calculates variance inflation factors to check for multicollinearity.

- **estat dwatson**

Calculates the Durbin–Watson test for first-order autocorrelation in time series (**tsset**) data. Chapter 12 gives examples of this and other time series procedures.

- **acprplot x1, mspline msopts(bands(7))**

Constructs an augmented component-plus-residual plot (also known as an augmented partial residual plot), often better than **cprplot** in screening for nonlinearities. The options **mspline msopts(bands(7))** call for connecting with line segments the cross-medians of seven vertical bands. Alternatively, we might ask for a lowess-smoothed curve with bandwidth 0.5 by specifying the options **lowess lsopts(bwidth(.5))**.

- **avplot x1**

Constructs an added-variable plot (also called a partial-regression or leverage plot) showing the relationship between *y* and *x1*, both adjusted for other *x* variables. Such plots help to notice outliers and influence points.

- . **avplots**
Draws and combines in one image all the added-variable plots from the recent **anova** or **regress**.
- . **cprplot x1**
Constructs a component-plus-residual plot (also known as a partial-residual plot) showing the adjusted relationship between *y* and predictor *x1*. Such plots help detect nonlinearities in the data.
- . **lvr2plot**
Constructs a leverage-versus-squared-residual plot (also known as an L–R plot).
- . **rvfplot**
Graphs the residuals versus the fitted (predicted) values of *y*.
- . **rvpplot x1**
Graphs the residuals against values of predictor *x1*.
- . **regress y x1 x2 i.catvar i.catvar#c.x2**
Performs regression of *y* on predictors *x1*, *x2*, a set of dummy variables created automatically to represent categories of *catvar*, and a set of interaction terms equal to those dummy variables times measurement (continuous) variable *x2*. **help fvvarlist** obtains more details.
- . **stepwise, pr(.05): regress y x1 x2 x3**
Performs stepwise regression using backward elimination until all remaining predictors are significant at the .05 level. All listed predictors are entered on the first iteration. Thereafter, each iteration drops one predictor with the highest *p* value, until all predictors remaining have probabilities below the probability to retain, *pr(.05)*. Options permit forward or hierarchical selection. The **stepwise** prefix works with many other model-fitting commands as well; type **help stepwise** for a list.
- . **regress y x1 x2 x3 [aweight = w]**
Performs weighted least squares (WLS) regression of *y* on *x1*, *x2* and *x3*. Variable *w* holds the analytical weights, which work as if we had multiplied each variable and the constant by the square root of *w*, and then performed an ordinary regression. Analytical weights are often employed to correct for heteroskedasticity when the *y* and *x* variables are means, rates or proportions, and *w* is the number of individuals making up each aggregate observation (e.g., city or school) in the data. If the *y* and *x* variables are individual-level, and the weights indicate numbers of replicated observations, then use frequency weights [**fweight = w**] instead. See **help survey** if the weights reflect design factors such as disproportionate sampling (Chapter 4).
- . **svy: regress y x1 x2 x3**
Performs survey weighted regression of *y* on *x1*, *x2* and *x3*. Assumes survey-type data previously identified by a **svyset** command (see Chapter 4).

Simple Regression

File *Nations2.dta* contains U.N. human-development indicators for 194 countries:

```
. use C:\data\Nations2.dta, clear
. describe country region life school chldmort adfert gdp
```

variable name	storage type	display format	value label	variable label
country	str21	%21s		Country
region	byte	%8.0g	region	Region
life	float	%9.0g		Life expectancy at birth 2005/2010
school	float	%9.0g		Mean years schooling (adults) 2005/2010
chldmort	float	%9.0g		Prob dying before age 5/1000 live births 2005/2009
adfert	float	%8.0g		Adolescent fertility: births/1000 fem 15-19, 2010
gdp	float	%9.0g		Gross domestic product per cap 2005\$, 2006/2009

```
. summarize life school chldmort adfert gdp
```

variable	Obs	Mean	Std. Dev.	Min	Max
life	194	68.7293	10.0554	45.85	82.76666
school	188	7.45922	2.959589	1.15	12.7
chldmort	193	47.65026	52.8094	2.25	209
adfert	194	51.81443	44.06612	1	207.1
gdp	179	12118.74	13942.34	279.8	74906

Life expectancies (*life*) exhibit much place-to-place variation. For example, Figure 7.1 shows that they tend to be lower in Africa than elsewhere.

```
. graph box life, over(region) marker(1, mlabel(country))
```

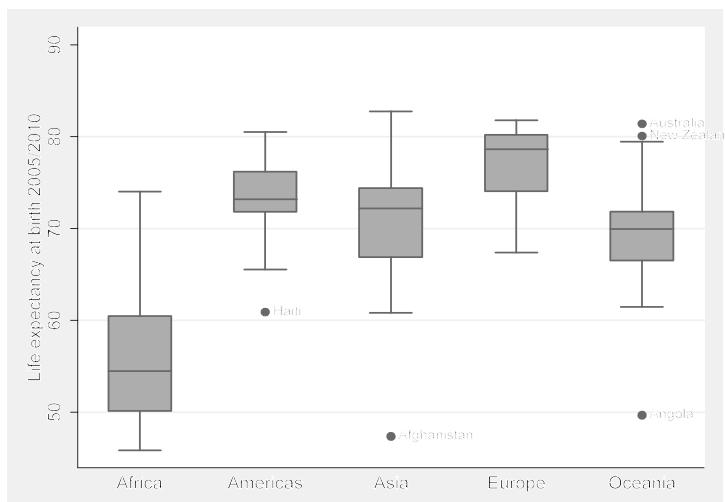


Figure 7.1

To what extent can variations in life expectancy be explained by average education, per capita wealth and other development indicators? We might begin to study education effects by conducting a simple regression of life expectancy on mean years of schooling. Basic syntax for Stata's regression command is **regress y x**, where *y* is the predicted or dependent variable, and *x* the predictor or independent variable.

. **regress life school**

Source	SS	df	MS	Number of obs = 188 F(1, 186) = 206.34 Prob > F = 0.0000 R-squared = 0.5259 Adj R-squared = 0.5234 Root MSE = 6.9079			
Model	9846.65406	1	9846.65406				
Residual	8875.86926	186	47.7197272				
Total	18722.5233	187	100.120446				
life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]		
school	2.45184	.1706856	14.36	0.000	2.115112	2.788569	
_cons	50.35941	1.36924	36.78	0.000	47.65817	53.06065	

As might be expected, life expectancies tend to be higher in countries with higher levels of schooling. A causal interpretation is premature at this point, but the regression table conveys information about the linear statistical relationship between *life* and *school*. At upper right it gives an overall *F* test, based on the sums of squares at the upper left. This *F* test evaluates the null hypothesis that coefficients on all *x* variables in the model (here there is only one *x* variable, *school*) equal zero. The *F* statistic, 206.34 with 1 and 186 degrees of freedom, leads easily to rejection of this null hypothesis ($p = .0000$ to four decimal places, meaning $p < .00005$). Prob > *F* means “the probability of a greater *F* statistic” if we drew many random samples from a population in which the null hypothesis is true.

At upper right, we also see the coefficient of determination, $R^2 = .5259$. Schooling explains about 53% of the variance in life expectancies. Adjusted R^2 , $R^2_a = .5234$, takes into account the complexity of the model relative to the complexity of the data.

The lower half of the regression table gives the fitted model itself. We find coefficients (slope and *y*-intercept) in the first column. The coefficient on *school* is 2.45184, and the *y*-intercept (listed as the coefficient on *_cons*) is 50.35941. Thus our regression equation is approximately predicted *life* = 50.36 + 2.45*school*

For every additional year of mean schooling, predicted life expectancy rises 2.45 years. This equation predicts a life expectancy of 50.36 years for a country where the mean schooling is zero — although the lowest value in the data is 1.15 years of schooling (Mozambique).

The second column lists estimated standard errors of the coefficients. These are used to calculate *t* tests (columns 3–4) and confidence intervals (columns 5–6) for each regression coefficient. The *t* statistics (coefficients divided by their standard errors) test null hypotheses that the corresponding population coefficients equal zero. At the $\alpha = .05$ or $.001$ significance levels, we could reject this null hypothesis regarding both the coefficient on *school* and the *y*-intercept; both probabilities show as “.000” (meaning $p < .0005$). Stata's modeling procedures ordinarily show 95% confidence intervals, but we can request other levels by specifying the **level()** option. For example, to see 99% confidence intervals instead, type

```
. regress life school, level(99)
```

After fitting a regression model, we could re-display the results just by typing **regress**, without arguments. Typing **regress, level(90)** would repeat the results but show 90% confidence intervals this time. Because the *Nations2.dta* dataset used in these examples does not represent a random sample from some larger population of nations, hypothesis tests and confidence intervals lack their literal meanings.

Mean years of schooling among these nations range from 1.15 to 12.7. What mean life expectancies does our model predict for nations with, for example, 2 or 12 years of schooling? The **margins** command offers a quick way to view predicted means along with their confidence intervals, and z tests (which often are not interesting) for whether those means differ from zero. A “vertical squish” **vsquish** option reduces the number of blank lines between rows in the table.

		Number of obs = 188			
Adjusted predictions					
Model VCE : OLS					
Expression : Linear prediction, predict()					
1._at	: school = 2				
2._at	: school = 12				
<hr/>					
		Delta-method			
Margin		Std. Err.	z	P> z	[95% Conf. Interval]
<hr/>					
_at					
1	55.26309	1.059291	52.17	0.000	53.18692
2	79.78149	.9244047	86.31	0.000	77.96969
					81.59329

At *school* = 2, predicted mean life expectancy equals 55.26 years, with a confidence interval from 53.19 to 57.34. At *school* = 12, predicted mean life expectancy is 79.78 years with an interval from 77.97 to 81.59. We could obtain predicted means of life expecting for *school* values at 1-year intervals from 2 through 12, and graph the results, by typing two commands:

```
. margins, at(school = (2(1)12)) vsquish
. marginsplot
```

In regression tables, the term *_cons* stands for the regression constant, usually set at one (so the coefficient on *_cons* equals the *y* intercept). Stata automatically includes a constant unless we tell it not to. A **nocons** option would cause Stata to suppress the constant, performing regression through the origin:

```
. regress y x, nocons
```

For some applications you might wish to specify your own constant. If the right-hand side variables include a user-supplied constant (named *c*, for example), employ the **hascons** option instead of **nocons**:

```
. regress y c x, hascons
```

Using **nocons** in this situation would result in a misleading F test and R^2 . Consult the *Base Reference Manual* or **help regress** for more about **hascons**.

Regression with one predictor amounts to finding a straight line that best fits the scatter of data, with “best fit” defined by the ordinary least squares (OLS) criterion. An easy way to graph this line is to draw a scatterplot (**twoway scatter**) overlaid with the a linear fit (**lfit**) plot. The command below would draw a basic version (not shown),

```
. graph twoway scatter life school || lfit life school
```

Figure 7.2 displays a nicer version, suppressing the unneeded legend, and inserting the regression equation as text. The variable names *life* and *school* illustrate how to italicize text in Stata graphs.

```
. graph twoway scatter life school || lfit life school
    || , legend(off) ytitle("Life expectancy in years")
    text(85 4 "predicted life = 50.36 + 2.45school")
```

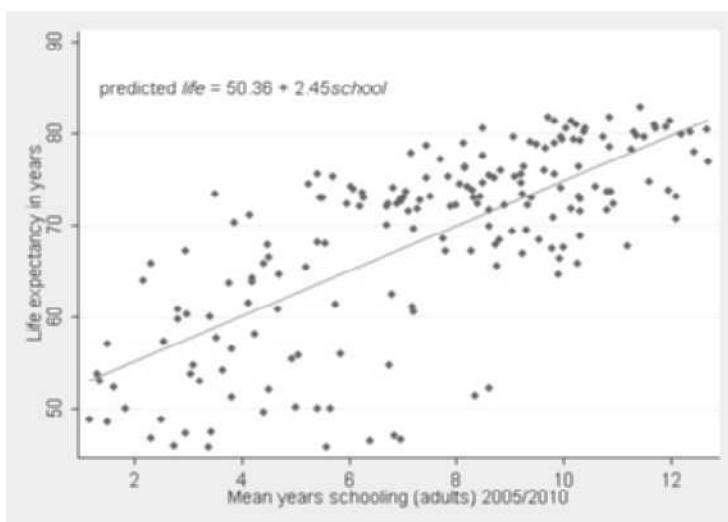


Figure 7.2

Correlation

Ordinary least-squares (OLS) regression finds the best-fitting straight line. A Pearson product-moment correlation coefficient describes how well the best-fitting line fits. **correlate** obtains correlations for listed variables.

```
. correlate gdp school adfert chdmort life
```

(obs=178)

	gdp	school	adfert	chldmort	life
gdp	1.0000				
school	0.5717	1.0000			
adfert	-0.5121	-0.6798	1.0000		
chldmort	-0.5160	-0.7724	0.7888	1.0000	
life	0.6062	0.7313	-0.7424	-0.9294	1.0000

correlate reports correlations based only on those observations that have non-missing values on all of the listed variables. From the table above we learn that only 178 of the 194 nations in *Nations2.dta* have complete information on all five of these variables. Those 178 correspond to the subset of observations that would be used in fitting models such as a multiple regression analysis that involves all of these variables.

Analysts not employing regression or other multi-variable techniques, however, might prefer to find correlations based on all of the observations available for each variable pair. The command **pwcorr** (pairwise correlation) accomplishes this. It can also furnish *t*-test probabilities for the null hypotheses that each individual correlation equals zero. In this example, the **star(.05)** option requests stars (*) marking correlations individually significant at the $\alpha = .05$ level.

```
. pwcorr gdp school adfert chldmort life, star(.05)
```

	gdp	school	adfert	chldmort	life
gdp	1.0000				
school	0.5733*	1.0000			
adfert	-0.5171*	-0.6752*	1.0000		
chldmort	-0.5160*	-0.7727*	0.7774*	1.0000	
life	0.6112*	0.7252*	-0.7318*	-0.9236*	1.0000

It is worth recalling, however, that if we drew many random samples from a population in which all variables really had zero correlations, about 5% of the sample correlations would nonetheless test “statistically significant” at the .05 level. Inexperienced analysts who review many individual hypothesis tests such as those in a **pwcorr** matrix, to identify the handful that are significant at the .05 level, run a much higher than .05 risk of making a Type I error. This problem is called the multiple-comparison fallacy. **pwcorr** offers two methods, Bonferroni and Šidák, for adjusting significance levels to take multiple comparisons into account. Of these, the Šidák method is more accurate. Significance-test probabilities are adjusted for the number of comparisons made.

```
. pwcorr gdp school adfert chldmort life, sidak sig star(.05)
```

	gdp	school	adfert	chldmort	life
gdp	1.0000				
school	0.5733*	1.0000			
	0.0000				
adfert	-0.5171*	-0.6752*	1.0000		
	0.0000	0.0000			
chldmort	-0.5160*	-0.7727*	0.7774*	1.0000	
	0.0000	0.0000	0.0000		
life	0.6112*	0.7252*	-0.7318*	-0.9236*	1.0000
	0.0000	0.0000	0.0000	0.0000	

The adjustments have minor effects with the moderate to strong correlations above, but could become critical with weaker correlations or more variables. In general, the more variables we correlate, the more the adjusted probabilities will exceed their unadjusted counterparts. See the *Base Reference Manual*'s discussion of **oneway** for the formulas involved.

Because Pearson correlations measure how well an OLS regression line fits, such correlations share the assumptions and weaknesses of OLS. Like OLS, correlations should generally not be interpreted without a look at the corresponding scatterplots. Scatterplot matrices provide a quick way to do this, using the same organization as a correlation matrix. Figure 3.12 in Chapter 3 shows a scatterplot matrix corresponding to the **pwcorr** commands above. By default, **graph matrix** applies pairwise deletion like **pwcorr**, so each small scatterplot shows all observations with nonmissing values on that particular pair of variables.

To obtain a scatterplot matrix corresponding to a **correlate** matrix or multiple regression, from which all observations having missing values are omitted, we qualify the command. One way to exclude observations with missing values on any of the listed variables employs the “not missing” (**!missing**) function (Figure 7.3):

```
. graph matrix gdp school adfert chldmort life
    if !missing(gdp,school,adfert,chldmort,life), half msymbol(+)
```

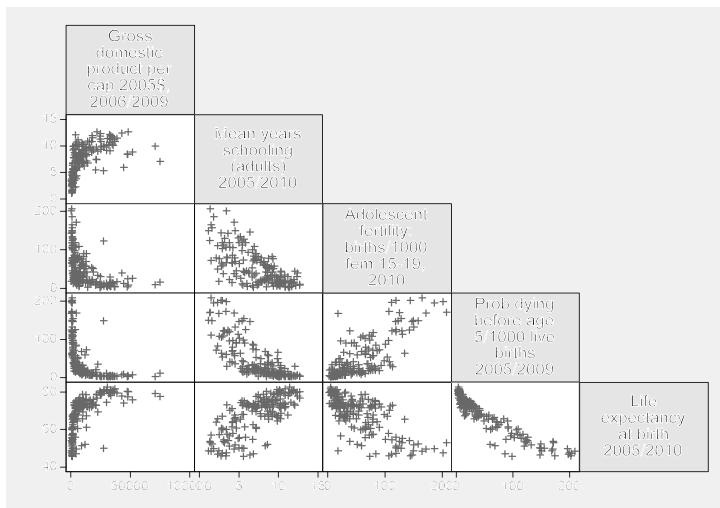


Figure 7.3

Figure 7.3 reveals what the correlation matrix does not: relationships involving per capita GDP are distinctly nonlinear. Consequently the correlation coefficients, or linear regression, provide poor descriptions of these relationships, and their significance tests are invalid.

Adding the **covariance** option after **correlate** produces a matrix of variances and covariances instead of correlations.

```
. correlate w x y z, covariance
```

Typing the following after a regression analysis displays the matrix of correlations between estimated coefficients, sometimes used to diagnose multicollinearity.

```
. estat vce, correlation
```

The following command will display the estimated coefficients' variance–covariance matrix, from which standard errors are derived.

```
. estat vce
```

In addition to Pearson correlations, Stata can also calculate several rank-based correlations. These can be employed to measure associations between ordinal variables, or as an outlier-resistant alternative to Pearson correlation for measurement variables. To obtain the Spearman rank correlation between *life* and *school*, equivalent to the Pearson correlation if these variables were transformed into ranks, type

```
. spearman life school
```

```

Number of obs =      188
Spearman's rho =     0.7145

Test of Ho: life and school are independent
Prob > |t| =      0.0000

```

Kendall's τ_a (tau-a) and τ_b (tau-b) rank correlations can be found easily for these data, although with larger datasets their calculation becomes slow:

```

. ktau life school

Number of obs =      188
Kendall's tau-a =     0.5142
Kendall's tau-b =     0.5149
Kendall's score =    9039
SE of score =      862.604  (corrected for ties)

Test of Ho: life and school are independent
Prob > |z| =      0.0000  (continuity corrected)

```

For comparison, here is the Pearson correlation with its unadjusted p -value:

```

. pwcorr life school, sig

          life   school
-----+
life       1.0000
-----+
school    0.7252   1.0000
           0.0000

```

In this example, both **spearman** (.71) and **pwcorr** (.73) yield higher correlations than **ktau** (.51). All three agree that null hypotheses of no association can be rejected.

Multiple Regression

Simple regression and correlation establish that a country's life expectancy is related to the mean years of schooling: *school* by itself explains about 52% of the variance of *life*. But might that relationship be spurious, occurring just because both schooling and life expectancy reflect a country's economic wealth? Does schooling matter once we control for regional variations? Are there additional factors besides schooling that can explain substantially more than 52% of the variance in life expectancy? Multiple regression addresses these sorts of questions.

We can incorporate other possible predictors of *life* simply by listing these variables in the **regress** command. For example, the following command would regress life expectancy on per capital GDP, adolescent fertility rate and child mortality.

```
. regress life school gdp adfert chdmort
```

Results from the above command are not shown because they would be misleading. We already know from Figure 7.3 that *gdp* exhibits distinctly nonlinear relationships with *life* and other variables in these data. We should work instead with a transformed version of *gdp* that exhibits

more linear relationships. Logarithms are an obvious and popular choice for transformation. After generating a new variable equal to the base-10 log of *gdp*, Figure 7.4 confirms that its relationships with other variables appear closer to linear, although some nonlinearity remains.

```
. generate loggdp = log10(gdp)
. label variable loggdp "log10(per cap GDP)"
. graph matrix gdp loggdp school adfert chldmort life
    if !missing(gdp,school,adfert,chldmort,life), half msymbol(dh)
```

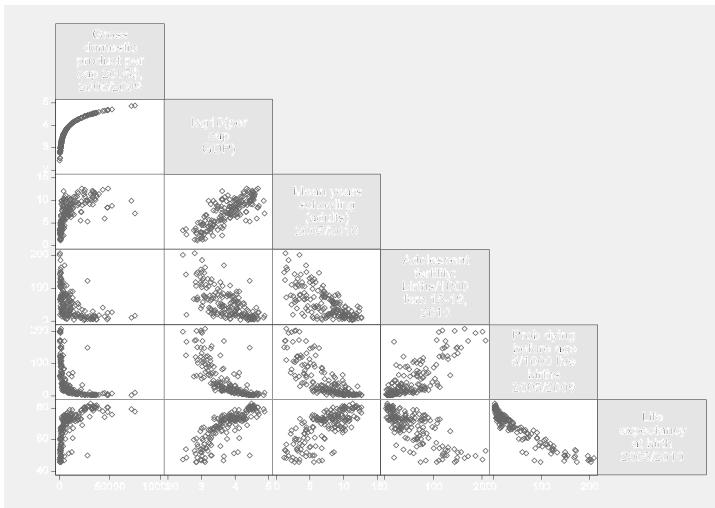


Figure 7.4

In Chapter 8 we will explore a different approach to transformation called Box–Cox regression. For the moment, let's consider *loggdp* to be good enough, and proceed with the regression example. Regressing life expectancy on schooling, log of GDP, adolescent fertility and child mortality rate yields a model that explains 88% of the variance in *life*.

```
. regress life school loggdp adfert chldmort
```

Source	SS	df	MS	Number of obs	=	178
Model	15545.2558	4	3886.31395	F(4,	173) = 321.24
Residual	2092.93402	173	12.0978845	Prob > F	=	0.0000
Total	17638.1898	177	99.6507898	R-squared	=	0.8813
				Adj R-squared	=	0.8786
				Root MSE	=	3.4782
life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
school	-.2339704	.1558288	-1.50	0.135	-.5415407	.0735999
loggdp	4.052938	.8133392	4.98	0.000	2.447592	5.658283
adfert	-.0004683	.0096839	-0.05	0.961	-.019582	.0186455
chldmort	-.1511827	.0098966	-15.28	0.000	-.1707163	-.131649
_cons	62.2544	3.114434	19.99	0.000	56.10722	68.40158

The multiple regression equation,

$$\text{predicted } life = 62.25 - .23\text{school} + 4.05\text{loggdp} - .00\text{ad fert} - .15\text{chldmort}$$

tells a much different story than the earlier simple regression,

$$\text{predicted } life = 50.36 + 2.45\text{school}$$

Once we control for three other variables, the coefficient on *school* becomes negative, and so much weaker (-.23 versus +2.45) that it no longer is statistically distinguishable from zero ($t = -1.50, p = .135$). Adolescent fertility has a coefficient only one-twentieth of one standard error from zero, which of course is not significant either ($t = -.05, p = .961$). *loggdp* and *chldmort* on the other hand show substantial, statistically significant effects. Life expectancy tends to be higher in wealthier countries with lower childhood mortality.

The coefficient on *chldmort* tells us that predicted life expectancy declines by .15 years with each 1-point rise in child mortality, if other predictors remain the same. The coefficient on *loggdp* indicates that, other things being equal, life expectancy rises by 4.05 years with each power-of-10 increase in per capita GDP. Per capita GDP varies in these data by more than two orders of magnitude, from \$279.8/person (Democratic Republic of the Congo) to \$74,906/person (Qatar).

The four predictors together explain about 88% of the variance in life expectancy ($R^2_a = .8786$). Adjusted R^2_a is the preferred summary statistic in multiple regression because unlike unadjusted R^2 ($R^2 = .8813$), R^2_a imposes a penalty for making models overly complex. R^2 will always go up when we add more predictors, but R^2_a may not.

The near-zero effect of *ad fert* and the weak, nonsignificant effect of *school* suggest that our four-predictor model is unnecessarily complex. Including irrelevant predictors in a model tends to inflate standard errors for other predictors, resulting in less precise estimates of their effects. A more parsimonious and efficient reduced model can be obtained by dropping nonsignificant predictors one at a time. First setting aside *ad fert*,

. regress life school loggdp chldmort

Source	SS	df	MS	Number of obs	=	178
Model	15545.2275	3	5181.7425	F(3, 174)	=	430.79
Residual	2092.9623	174	12.028519	Prob > F	=	0.0000
Total	17638.1898	177	99.6507898	R-squared	=	0.8813
				Adj R-squared	=	0.8793
				Root MSE	=	3.4682
life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
school	-.233002	.1540928	-1.51	0.132	-.5371337	.0711297
loggdp	4.056047	.808465	5.02	0.000	2.460387	5.651708
chldmort	-.1514263	.008493	-17.83	0.000	-.168189	-.1346637
_cons	62.22201	3.032798	20.52	0.000	56.2362	68.20782

and then leaving out *school*,

. regress life loggdp chldmort

Source	SS	df	MS	Number of obs = 178		
Model	15517.7253	2	7758.86267	F(2, 175) =	640.33	
Residual	2120.46446	175	12.1169398	Prob > F =	0.0000	
Total	17638.1898	177	99.6507898	R-squared =	0.8798	
				Adj R-squared =	0.8784	
				Root MSE =	3.4809	
life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
loggdp	3.510749	.7262322	4.83	0.000	2.077448	4.94405
chldmort	-.1457805	.0076563	-19.04	0.000	-.160891	-.13067
_cons	62.28614	3.043627	20.46	0.000	56.2792	68.29308

We end up with a reduced model having only two predictors, smaller standard errors, and virtually the same adjusted R^2 (.8784 with two predictors, or .8786 with four). The coefficient on *loggdp* ends up slightly lower, while that on *chldmort* remains almost unchanged.

$$\text{predicted life} = 62.29 + 3.51\log gdp - .15chldmort$$

We could calculate predicted values for any combination of *loggdp* and *chldmort* by substituting those values into the regression equation. The **margins** command obtains predicted means (also called adjusted means) of the dependent variable at specified values of one or more independent variables. For example, to see the predicted mean life expectancy, adjusted for *loggdp*, at *chldmort* values of 2, 100 and 200:

. margins, at(chldmort = (2 100 200)) vsquish			Number of obs = 178		
Predictive margins					
Model VCE : OLS					
Expression	: Linear prediction, predict()				
1._at	: chldmort	= 2			
2._at	: chldmort	= 100			
3._at	: chldmort	= 200			
	Delta-method			[95% Conf. Interval]	
	Margin	Std. Err.	z	P> z	
-at					
1	75.23421	.4408642	170.65	0.000	74.37013 76.09828
2	60.94772	.4733418	128.76	0.000	60.01998 61.87545
3	46.36966	1.189535	38.98	0.000	44.03822 48.70111

This **margins** table tells us that the predicted mean life expectancy, at *chldmort* = 2 across all observed values of *loggdp*, is 75.23. Similarly, the predicted mean life expectancy at *chldmort* = 200 is 46.37. If we include the **atmeans** option, *loggdp* would be set to its mean — giving an equivalent result in this instance. The output would then be labeled “adjusted predictions” instead of “predictive margins.”

We could also ask for predicted means for specified values of both *chldmort* and *loggdp*. The following commands request means at six combinations of values: *chldmort* at 2, 100 or 200, and *loggdp* at 2.5 or 4.5.

```
. margins, at(chldmort = (2 100 200) loggdp = (2.5 4.5)) vsquish
Adjusted predictions                               Number of obs = 178
Model VCE   : OLS

Expression : Linear prediction, predict()
1._at     : loggdp      = 2.5
             chldmort    = 2
2._at     : loggdp      = 2.5
             chldmort    = 100
3._at    : loggdp      = 2.5
             chldmort    = 200
4._at    : loggdp      = 4.5
             chldmort    = 2
5._at    : loggdp      = 4.5
             chldmort    = 100
6._at    : loggdp      = 4.5
             chldmort    = 200
```

	Delta-method					
	Margin	Std. Err.	z	P> z	[95% Conf. Interval]	
-at						
1	70.77145	1.244946	56.85	0.000	68.3314	73.2115
2	56.48496	.718987	78.56	0.000	55.07577	57.89415
3	41.90691	.7896567	53.07	0.000	40.35921	43.45461
4	77.79295	.4312218	180.40	0.000	76.94777	78.63813
5	63.50646	.9082472	69.92	0.000	61.72633	65.28659
6	48.92841	1.624056	30.13	0.000	45.74532	52.1115

The follow-up command **marginsplot** graphs **margins** results, in Figure 7.5.

```
. marginsplot
```

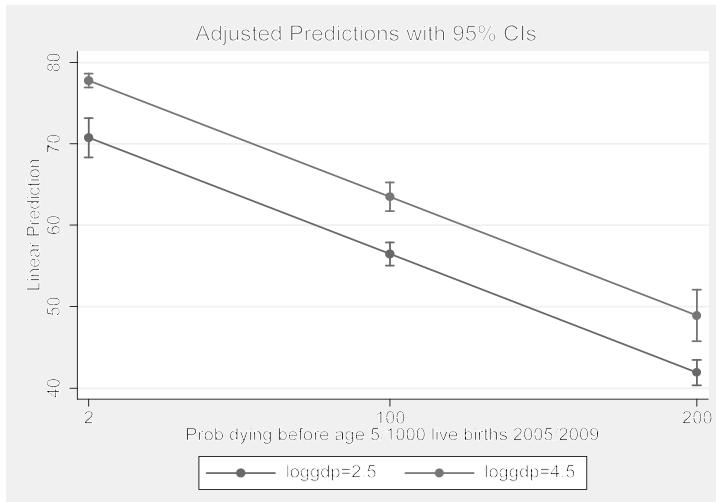


Figure 7.5

To obtain standardized regression coefficients (beta weights) with any regression, add the **beta** option. Standardized coefficients are what we would see in a regression where all the variables have been transformed into standard scores, which have means equal to 0 and standard deviations equal to 1.

```
. regress life loggdp chldmort, beta
```

Source	SS	df	MS	Number of obs	=	178
Model	15517.7253	2	7758.86267	F(2, 175)	=	640.33
Residual	2120.46446	175	12.1169398	Prob > F	=	0.0000
Total	17638.1898	177	99.6507898	R-squared	=	0.8798
				Adj R-squared	=	0.8784
				Root MSE	=	3.4809

life	Coef.	Std. Err.	t	P> t	Beta
loggdp	3.510749	.7262322	4.83	0.000	.1974935
chldmort	-.1457805	.0076563	-19.04	0.000	-.7778774
_cons	62.28614	3.043627	20.46	0.000	.

The standardized regression equation is

$$\text{predicted } life^* = .197 \text{loggdp}^* -.778 \text{chldmort}^*$$

where *life**, *loggdp** and *chldmort** denote these variables in standard-score form. For example, we could interpret the standardized coefficient on *chldmort* as follows:

$b_2^* = -.778$: Predicted life expectancy declines by .778 standard deviations, with each one-standard-deviation increase in the child mortality rate—if *loggdp* does not change.

The *F* and *t* tests, R^2 , and other aspects of the regression remain the same.

Hypothesis Tests

Two types of hypothesis tests appear in **regress** output tables. As with other common hypothesis tests, they begin from the assumption that observations in the sample at hand were drawn randomly and independently from an infinitely large population.

1. Overall *F* test: The *F* statistic at the upper right in the regression table evaluates the null hypothesis that in the population, coefficients on all the model's *x* variables equal zero.
2. Individual *t* tests: The third and fourth columns of the regression table contain *t* tests for each individual regression coefficient. These evaluate the null hypotheses that in the population, the coefficient on each particular *x* variable equals zero.

The *t* test probabilities are two-sided. For one-sided tests, divide these *p*-values in half.

In addition to these standard *F* and *t* tests, Stata can perform *F* tests of user-specified hypotheses. The **test** command refers back to the most recently fitted model such as **anova** or **regress**. Returning to our four-predictor regression example, suppose we wish to test the null hypothesis that both *ad fert* and *chldmort* (considered jointly) have zero effect.

```
. regress life school loggdp adfert chldmort
```

Source	SS	df	MS	Number of obs	=	178
Model	15545.2558	4	3886.31395	F(4, 173)	=	321.24
Residual	2092.93402	173	12.0978845	Prob > F	=	0.0000
Total	17638.1898	177	99.6507898	R-squared	=	0.8813
				Adj R-squared	=	0.8786
				Root MSE	=	3.4782

life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
school	-.2339704	.1558288	-1.50	0.135	-.5415407 .0735999
loggdp	4.052938	.8133392	4.98	0.000	2.447592 5.658283
adfert	-.0004683	.0096839	-0.05	0.961	-.019582 .0186455
chldmort	-.1511827	.0098966	-15.28	0.000	-.1707163 -.131649
_cons	62.2544	3.114434	19.99	0.000	56.10722 68.40158

```
. test adfert chldmort
```

```
( 1) adfert = 0
( 2) chldmort = 0

F( 2, 173) = 158.03
Prob > F = 0.0000
```

While the individual null hypotheses point in opposite directions (effect of *chldmort* significant, *adfert* not), the joint hypothesis that coefficients on *chldmort* and *adfert* both equal zero can reasonably be rejected ($p < .00005$). Such tests on subsets of coefficients are useful when we have several conceptually related predictors or when individual coefficient estimates appear unreliable due to multicollinearity.

test could duplicate the overall *F* test:

```
. test school loggdp adfert chldmort

( 1) school = 0
( 2) loggdp = 0
( 3) adfert = 0
( 4) chldmort = 0

F( 4, 173) = 321.24
Prob > F = 0.0000
```

test also could duplicate the individual-coefficient tests. Regarding the coefficient on *school*, for example, the *F* statistic obtained by **test** equals the square of the *t* statistic in the regression table, $2.25 = (-1.50)^2$, and yields exactly the same *p*-value:

```
. test school

( 1) school = 0

F( 1, 173) = 2.25
Prob > F = 0.1351
```

Applications of **test** more useful in advanced work (although not meaningful for the life-expectancy example at hand) include the following.

1. Test whether a coefficient equals a specified constant. For example, to test the null hypothesis that the coefficient on *school* equals 1 ($H_0: \beta_1 = 1$), instead of testing the usual null hypothesis that it equals 0 ($H_0: \beta_1 = 0$), type

```
. test school = 1
```

2. Test whether two coefficients are equal. For example, the following command evaluates the null hypothesis $H_0: \beta_2 = \beta_3$

```
. test loggdp = adfert
```

3. Finally, **test** understands some algebraic expressions. We could request something like the following, which would test $H_0: \beta_2 = (\beta_3 + \beta_4) / 100$

```
. test school = (loggdp + adfert)/100
```

Consult **help test** for more information and examples.

Dummy Variables

Categorical variables can become predictors in a regression when they are expressed as one or more {0,1} dichotomies called dummy variables. For example, we have already seen large regional differences in life expectancies (Figure 7.1). The categorical variable *region* takes values from 1 (Africa) to 5 (Oceania) which can be re-expressed as a set of five {0,1} dummy variables. The **tabulate** command offers an automatic way to do this, generating one dummy variable for each category of the tabulated variable when we include a **gen** (generate) option. In the example below the resulting dummy variables are named *reg1* through *reg5*. *reg1* equals 1 for African nations and 0 for others; *reg2* equals 1 for the Americas and 0 for others; and so forth.

```
. tabulate region, gen(reg)
```

Region	Freq.	Percent	Cum.
Africa	52	26.80	26.80
Americas	35	18.04	44.85
Asia	49	25.26	70.10
Europe	43	22.16	92.27
Oceania	15	7.73	100.00
Total	194	100.00	

```
. describe reg*
```

variable name	storage type	display format	value label	variable label
region	byte	%8.0g	region	Region
reg1	byte	%8.0g	region==Africa	region==Africa
reg2	byte	%8.0g	region==Americas	region==Americas
reg3	byte	%8.0g	region==Asia	region==Asia
reg4	byte	%8.0g	region==Europe	region==Europe
reg5	byte	%8.0g	region==Oceania	region==Oceania

```
. label values reg1 reg1
. label define reg1 0 "others" 1 "Africa"
. tabulate reg1 region
```

region==Africa	Region					Total
	Africa	Americas	Asia	Europe	Oceania	
others	0	35	49	43	15	142
Africa	52	0	0	0	0	52
Total	52	35	49	43	15	194

Regressing *life* on one dummy variable, *reg1* (Africa), is equivalent to performing a two-sample *t* test of whether mean *life* is the same across categories of *reg1*. Is the mean life expectancy significantly different, comparing Africa with other parts of the world?

```
. ttest life, by(reg1)
```

Two-sample t test with equal variances

Group	obs	Mean	Std. Err.	Std. Dev.	[95% Conf. Interval]
others	142	73.21115	.5068244	6.03951	72.20919 74.21311
Africa	52	56.49038	1.185937	8.551912	54.10952 58.87125
combined	194	68.7293	.7219359	10.0554	67.3054 70.15319
diff		16.72077	1.101891		14.5474 18.89413

```
diff = mean(others) - mean(Africa)          t = 15.1746
Ho: diff = 0                                degrees of freedom = 192
                                             
Ha: diff < 0      Ha: diff != 0      Ha: diff > 0
Pr(T < t) = 1.0000    Pr(|T| > |t|) = 0.0000    Pr(T > t) = 0.0000
```

```
. regress life reg1
```

Source	SS	df	MS	Number of obs	=	194
Model	10641.4858	1	10641.4858	F(1, 192)	=	230.27
Residual	8872.96636	192	46.2133664	Prob > F	=	0.0000
Total	19514.4521	193	101.111151	R-squared	=	0.5453
				Adj R-squared	=	0.5429
				Root MSE	=	6.798
life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
reg1	-16.72077	1.101891	-15.17	0.000	-18.89413	-14.5474
_cons	73.21115	.570479	128.33	0.000	72.08594	74.33636

The *t* test confirms that the 16.72-year difference between means for Africa (56.49) and other regions (73.21) is statistically significant (*t* = 15.17, *p* = .000). We get exactly the same result from the dummy variable regression (*t* = -15.17, *p* = .000), where the coefficient *reg1* ($b_1 = -16.72$) likewise indicates that mean life expectancy is 16.72 years lower in Africa than in other regions ($b_0 = 73.21$).

Figure 7.6 graphs this dummy variable regression. All the data points line up along two vertical bands at *reg1* = 1 (Africa) and *reg1* = 0 (elsewhere). To spread the points out visually this graph example employs a jitter(5) option, which adds a small amount of spherical random noise to the location of each point, so they do not all plot on top of each other. jitter() does not affect the regression line, which simply connects the mean of *life* when *reg1* = 0 (73.21) with

the mean of *life* when *reg1* = 1 (56.49). Both of these means or predicted values are plotted as solid squares. The difference between the two means equals the regression slope, -16.72 years. Note that the 0 and 1 values of *reg1* are re-labeled in the **xlabel()** option of this **graph** command.

```
. predict lifehat
. graph twoway scatter life lifehat reg1, msymbol(oh S) jitter(5)
    || lfit life reg1
    || , legend(off) xlabel(0 "others" 1 "Africa")
    xtitle("Global region") ytitle("Life expectancy in years")
```

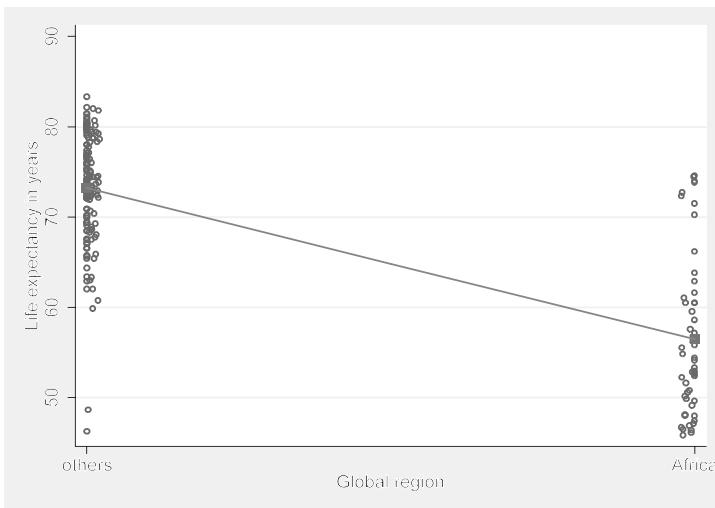


Figure 7.6

The five world regions have been re-expressed as five dummy variables, but it is not possible to include all five in one regression because of multicollinearity: the values of any four of these dummy variables perfectly determine the fifth. Consequently, we can represent all the information of a k -category categorical variable through $k-1$ dummy variables. For example, we earlier saw that per capita gross domestic product (in log form, *loggdp*) and child mortality rate (*chldmort*) together explain about 88% of the variance in life expectancy. Including four dummy variables for regions 1–4 raises this only to about 89% ($R^2_a = .8872$).

```
. regress life reg1 reg2 reg3 reg4 loggdp chldmort
```

Source	SS	df	MS	Number of obs = 178 F(6, 171) = 233.00 Prob > F = 0.0000 R-squared = 0.8910 Adj R-squared = 0.8872 Root MSE = 3.3529		
Model	15715.8742	6	2619.31237			
Residual	1922.31561	171	11.2416118			
Total	17638.1898	177	99.6507898			
life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
reg1	-2.150794	1.211918	-1.77	0.078	-4.543041	.2414527
reg2	1.486722	1.176501	1.26	0.208	-.8356127	3.809057
reg3	.9838334	1.129945	0.87	0.385	-1.246603	3.21427
reg4	1.455465	1.199846	1.21	0.227	-.9129513	3.823882
loggdp	3.239467	.7342834	4.41	0.000	1.79004	4.688894
chldmort	-.1270253	.0086971	-14.61	0.000	-.1441928	-.1098578
_cons	62.16206	3.10137	20.04	0.000	56.04016	68.28396

None of the regional dummy variables have significant effects, when we include them all and control for *loggdp* and *chldmort*. The nonsignificant coefficients suggests that a simpler model might fit just as well, and give a clearer picture of those effects that really do matter. The first step toward a reduced model involves dropping *reg3*, the weakest of these predictors. The result below fits just as well ($R^2_a = .8873$) and yields more precise estimates (lower standard errors) of other region effects. The coefficient on *reg1* now appears significant.

```
. regress life reg1 reg2 reg4 loggdp chldmort
```

Source	SS	df	MS	Number of obs = 178 F(5, 172) = 279.84 Prob > F = 0.0000 R-squared = 0.8905 Adj R-squared = 0.8873 Root MSE = 3.3505		
Model	15707.3519	5	3141.47038			
Residual	1930.83792	172	11.2258018			
Total	17638.1898	177	99.6507898			
life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
reg1	-2.927382	.8199249	-3.57	0.000	-4.545793	-1.308972
reg2	.6920922	.7419319	0.93	0.352	-.7723717	2.156556
reg4	.6487658	.7618415	0.85	0.396	-.8549968	2.152528
loggdp	3.273944	.7326992	4.47	0.000	1.827705	4.720184
chldmort	-.1269767	.0086908	-14.61	0.000	-.1441311	-.1098224
_cons	62.82061	3.00561	20.90	0.000	56.88798	68.75324

Next, dropping *reg4* and finally *reg2* results in a reduced model that still explains 89% of the variance in life expectancy ($R^2_a = .8879$) but with just three predictors.

```
. regress life reg1 loggdp chldmort
```

Source	SS	df	MS			
Model	15694.5388	3	5231.51293	Number of obs =	178	
Residual	1943.65102	174	11.1704082	F(3, 174) =	468.34	
Total	17638.1898	177	99.6507898	Prob > F =	0.0000	
				R-squared =	0.8898	
				Adj R-squared =	0.8879	
				Root MSE =	3.3422	
life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
reg1	-3.143763	.7901811	-3.98	0.000	-4.703336	-1.584189
loggdp	3.414611	.6977087	4.89	0.000	2.037549	4.791672
chldmort	-.1277141	.0086406	-14.78	0.000	-.144768	-.1106603
_cons	62.65707	2.923818	21.43	0.000	56.88636	68.42779

From this purely statistical investigation we might conclude that the differences in life expectancy among other regions of the world are largely accounted for by variations in wealth and child mortality, but in Africa there are circumstances at work (such as wars) that further depress life expectancy.

Interaction Effects

The previous section described what are called “intercept dummy variables,” because their coefficients amount to shifts in a regression equation’s *y* intercept, comparing the 0 and 1 groups. Another use for dummy variables is to form interaction terms called “slope dummy variables” by multiplying a dummy times a measurement variable. In this section we stay with the *Nations2.dta* data, but consider some different variables: per capita carbon dioxide emissions (*co2*), percent of the population living in urban areas (*urban*), and the dummy variable *reg4* defined as 1 for European countries and 0 for all others. We start out by labeling the values of *reg4*, and calculating a log version of *co2* because of that variable’s severe positive skew.

```
. label values reg4 reg4
. label define reg4 0 "others" 1 "Europe"
. generate logco2 = log10(co2)
. label variable logco2 "log10(per cap CO2)"
. describe urban reg4 co2 logco2
```

variable	storage type	display format	value label	variable label
urban	float	%9.0g		Percent population urban 2005/2010
reg4	byte	%8.0g	reg4	region==Europe
co2	float	%9.0g		Tons of CO2 emitted per cap 2005/2006
logco2	float	%9.0g		log10(per cap CO2)

We form an interaction term or slope dummy variable named *urb_reg4* by multiplying the dummy variable *reg4* times the measurement variable *urban*. The resulting variable *urb_reg4* equals *urban* for countries in Europe, and zero for all other countries.

```
. generate urb_reg4 = urban * reg4
. label variable urb_reg4 "interaction urban*reg4 (Europe)"
```

Regressing *logco2* on *urban*, *reg4* and the interaction term *urb_reg4* gives a model that tests whether either the y-intercept or the slope relating *logco2* to *urban* might be different for Europe compared with other regions.

```
. regress logco2 urban reg4 urb_reg4
```

Source	SS	df	MS	Number of obs =	185
Model	55.8882644	3	18.6294215	F(3, 181) =	72.86
Residual	46.2772694	181	.255675521	Prob > F =	0.0000
Total	102.165534	184	.555247466	R-squared =	0.5470
				Adj R-squared =	0.5395
				Root MSE =	.50564

logco2	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
urban	.0217385	.0017762	12.24	0.000	.0182339 .0252431
reg4	1.294163	.462044	2.80	0.006	.3824774 2.205848
urb_reg4	-.0133405	.0065573	-2.03	0.043	-.0262791 -.0004019
_cons	-.4682452	.1007257	-4.65	0.000	-.6669929 -.2694975

The interaction effect is statistically significant ($p = .043$), suggesting that the relationship between percent urban and log CO₂ emissions is different for Europe than for the rest of the world. The main effect of *urban* is positive (.0217), meaning that *logco2* tends to be higher in countries with more urbanized population. But the interaction coefficient is negative, meaning that this upward slope is less steep for Europe. We could write the model above as two separate equations:

Europe, *reg4* = 1:

$$\begin{aligned} \text{predicted } \logco2 &= -.4682 + .0217\text{urban} + 1.2942(1) - .0133\text{urban}(1) \\ &= -.4682 + 1.2942 + (.0217 - .0133)\text{urban} \\ &= .826 + .0084\text{urban} \end{aligned}$$

others, *reg4* = 0:

$$\begin{aligned} \text{predicted } \logco2 &= -.4682 + .0217\text{urban} + 1.2942(0) - .0133\text{urban}(0) \\ &= -.4682 + .0217\text{urban} \end{aligned}$$

The post-regression command **predict newvar** generates a new variable holding predicted values from the recent regression. Graphing the predicted values for this example visualizes our interaction effect (Figure 7.7). The line in the left-hand (*reg4* = 0) panel has a slope of .0217 and y-intercept -.4682. The line in the right panel (*reg4* = 1) has a less-steep slope (.0084) and a higher y-intercept (.826). No European countries exhibit the low-urbanization, low-CO₂ profile seen in other parts of the world; and even European nations with middling urbanization have relatively high CO₂ emissions.

```
. predict co2hat
. graph twoway scatter logco2 urban, msymbol(Oh)
    || connect co2hat urban, msymbol(+)
    || , by(reg4)
```

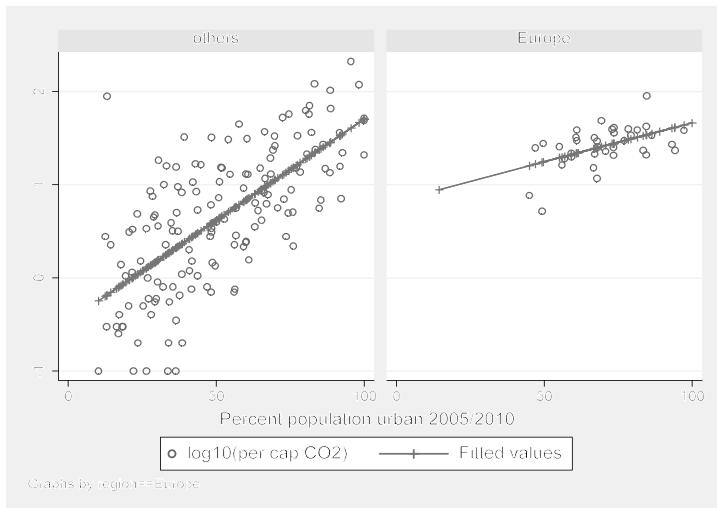


Figure 7.7

The **i.varname** and **c.varname** notation for indicator and continuous variables, introduced in Chapter 6, provides an alternative way to include interactions. The symbol **#** specifies an interaction between two variables, and **##** a factorial interaction which automatically includes all the lower-level interactions involving those variables. **reg4** is an indicator variable and **urban** is continuous, so the same model estimated above could be obtained by the command

```
. regress logco2 c.urban i.reg4 c.urban##i.reg4
```

or equivalently with a factor interaction,

```
. regress logco2 c.urban##i.reg4
```

Source	SS	df	MS	Number of obs = 185		
Model	55.8882644	3	18.6294215	F(3, 181)	=	72.86
Residual	46.2772694	181	.255675521	Prob > F	=	0.0000
Total	102.165534	184	.555247466	R-squared	=	0.5470
				Adj R-squared	=	0.5395
				Root MSE	=	.50564

logco2	coef.	Std. Err.	t	P> t	[95% Conf. Interval]
urban	.0217385	.0017762	12.24	0.000	.0182339 .0252431
1.reg4	1.294163	.462044	2.80	0.006	.3824774 2.205848
reg4#c.urban	-.0133405	.0065573	-2.03	0.043	-.0262791 -.0004019
_cons	-.4682452	.1007257	-4.65	0.000	-.6669929 -.2694975

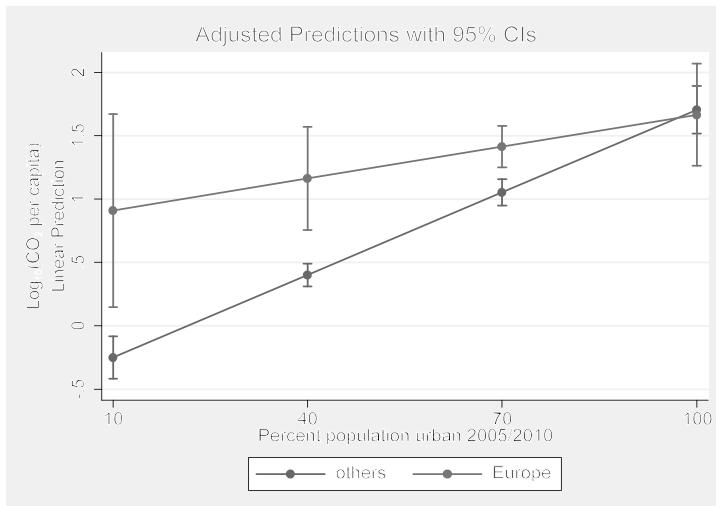
margins understands # or ## interactions. Percent urban ranges from about 10 to 100 percent in these data, so we could graph the interaction as follows. First calculate the predicted means of *logco2* at several combinations of *urban* (10, 40, 70 or 100) and *reg4* (0 or 1).

. margins, at(urban = (10(30)100) reg4 = (0 1)) vsquish									
Adjusted predictions				Number of obs = 185					
Model VCE : OLS									
Expression : Linear prediction, predict()									
1._at	: urban	=	10						
	reg4	=	0						
2._at	: urban	=	10						
	reg4	=	1						
3._at	: urban	=	40						
	reg4	=	0						
4._at	: urban	=	40						
	reg4	=	1						
5._at	: urban	=	70						
	reg4	=	0						
6._at	: urban	=	70						
	reg4	=	1						
7._at	: urban	=	100						
	reg4	=	0						
8._at	: urban	=	100						
	reg4	=	1						

	Delta-method					
	Margin	Std. Err.	z	P> z	[95% Conf. Interval]	
_at						
1	-.2508599	.084864	-2.96	0.003	-.4171903	-.0845296
2	.9098981	.3890654	2.34	0.019	.147344	1.672452
3	.4012958	.046435	8.64	0.000	.3102849	.4923067
4	1.161839	.2080449	5.58	0.000	.7540788	1.5696
5	1.053452	.052811	19.95	0.000	.9499438	1.156959
6	1.413781	.0831383	17.01	0.000	1.250833	1.576729
7	1.705607	.0953954	17.88	0.000	1.518636	1.892579
8	1.665722	.2055716	8.10	0.000	1.262809	2.068635

Next, use **marginsplot** to graph these means. Note the use of **twoway**-type options to label details of the graph. Figure 7.8 visualizes the same model as Figure 7.7, but in a different style that shows confidence intervals for the predicted means instead of data points. The option in the following **marginsplot** command specifies l2 (letter “el” 2), a second left-hand axis title.

```
. marginsplot, l2("Log{subscript:10}(CO{subscript:2}
    per capita)") xlabel(10(30)100)
```

**Figure 7.8**

Interaction effects could also involve two measurement variables. A trick called centering helps to reduce multicollinearity problems with such interactions, and makes their main effects easier to interpret. Centering involves subtracting their respective means from both variables before defining an interaction term as their product. Centered variables have means approximately equal to zero, and are negative for below-average values. The commands below calculate centered versions of *urban* and *loggdp*, named *urban0* and *loggdp0*. The interaction term *urb_gdp* is then defined as the product of *urban0* times *loggdp0*.

```
. summarize urban loggdp
    Variable   Obs      Mean    Std. Dev.     Min      Max
    urban       194    55.43488   23.4391    10.25    100
    loggdp      179    3.775729   .5632902   2.446848  4.874516

. generate urban0 = urban - 55.4
. label variable urban0 "Percent urban, centered"
. generate loggdp0 = loggdp - 3.78
. label variable loggdp0 "log10(GDP per cap), centered"
. generate urb_gdp = urban0 * loggdp0
. label variable urb_gdp "interaction urban0*loggdp0"
```

More precise centering could be performed using means returned by **summarize**:

```
. summarize urban
. generate urban00 = urban - r(mean)
```

We might also set aside all observations with missing values on any variables in the regression, before obtaining a mean for centering.

Regressing *logco2* on the centered main effects *loggdp0* and *urban0*, along with the interaction term *urb_gdp*, we find that the interaction effect is negative and statistically significant.

```
. regress logco2 loggdp0 urban0 urb_gdp
```

Source	SS	df	MS	Number of obs =	175
Model	83.4990753	3	27.8330251	F(3, 171) =	371.66
Residual	12.806051	171	.074889187	Prob > F =	0.0000
Total	96.3051263	174	.553477737	R-squared =	0.8670
				Adj R-squared =	0.8647
				Root MSE =	.27366

logco2	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
loggdp0	1.116759	.0558107	20.01	0.000	1.006592 1.226925
urban0	.0024787	.0013689	1.81	0.072	-.0002235 .0051809
urb_gdp	-.0082808	.0017418	-4.75	0.000	-.011719 -.0048425
_cons	.8950411	.0267376	33.47	0.000	.8422628 .9478194

The same regression could have been conducted, and the continuous-by-continuous variable interaction effect graphed, with the following three commands (results not shown).

```
. regress logco2 c.loggdp0 c.urban0 c.loggdp0#c.urban0
. margins, at(loggdp0 = (-1.3 1.1) urban0 =(-45 45))
. marginsplot
```

Main effects in a regression of this sort, where the interacting variables have been centered, can be interpreted as the effect of each variable when the other is at its mean. Thus, predicted *logco2* rises by 1.12 with each 1-unit increase in *loggdp*, when *urban* is at its mean. Similarly, predicted *logco2* rises by only a small amount, .0025, with each 1-unit increase in *urban* when *loggdp* is at its mean. The coefficient on interaction term *urb_gdp*, however, tells us that with each 1-unit increase in urbanization, the effect of *loggdp* on *logco2* becomes weaker, decreasing by −.008. That is, CO₂ emissions increase as wealth increases, but do so less steeply in more urbanized countries.

Robust Estimates of Variance

The standard errors and hypothesis tests that accompany ordinary regression (such as **regress** or **anova**) assume that errors follow independent and identical distributions. If this assumption is untrue, those standard errors probably will underestimate the true sample-to-sample variation, and yield unrealistically narrow confidence intervals or too-low test probabilities. To cope with a common problem called heteroskedasticity, **regress** and some other model fitting commands have an option that estimates standard errors without relying on the strong and sometimes implausible assumptions of independent, identically distributed errors. This option uses an approach derived independently by Huber, White and others that is sometimes referred to as a sandwich estimator of variance. Type **help vce option**, or see **vce_option** in the *Stata Reference Manual*, for technical details.

The previous section ended with a regression of *logco2* on three predictors: *loggdp0*, *urban0* and their product *urb_gdp*. To repeat this same regression but with robust standard errors, we just add the **vce(robust)** option:

```
. regress logco2 loggdp0 urban0 urb_gdp, vce(robust)

Linear regression                                         Number of obs =      175
                                                          F(  3,    171) =   410.66
Prob > F                                                 =  0.0000
R-squared                                                 =  0.8670
Root MSE                                                 = .27366
```

<i>logco2</i>	Coef.	Robust Std. Err.	t	P> t	[95% Conf. Interval]
<i>loggdp0</i>	1.116759	.0525277	21.26	0.000	1.013072 1.220445
<i>urban0</i>	.0024787	.0013123	1.89	0.061	-.0001116 .005069
<i>urb_gdp</i>	-.0082808	.0016976	-4.88	0.000	-.0116317 -.0049298
_cons	.8950411	.0271616	32.95	0.000	.8414258 .9486564

Descriptive aspects of the regression — the coefficients and R^2 — are identical with or without robust standard errors. On the other hand the robust standard errors themselves, along with confidence intervals, *t* and *F* tests, differ from their non-robust counterparts seen earlier. The differences here are slight, however. The basic results for this example do not depend on assuming errors that are independent and identically distributed across values of predictors.

The rationale underlying these robust standard-error estimates is explained in the *User's Guide*. Briefly, we give up on the classical goal of estimating true population parameters (β 's) for a model such as

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Instead, we pursue the less ambitious goal of simply estimating the sample-to-sample variation that our *b* coefficients might have, if we drew many random samples and applied OLS repeatedly to calculate *b* values for a model such as

$$y_i = b_0 + b_1 x_i + e_i$$

We do not assume that these *b* estimates will converge on some "true" population parameter. Confidence intervals formed using the robust standard errors therefore lack the classical interpretation of having a certain probability (across repeated sampling) of containing the true value of β . Rather, the robust confidence intervals have a certain probability (across repeated sampling) of containing *b*, defined as the value upon which sample *b* estimates converge. Thus, we pay for relaxing the identically-distributed-errors assumption by settling for a less impressive conclusion.

Another robust-variance option, **vce(cluster *clustervar*)**, allows us to relax the independent-errors assumption in a limited way, when errors are correlated within subgroups or clusters of the data. For example, in the cross-national data we have seen substantial differences in variation by region. Adding the option **vce(cluster *region*)** obtains robust standard errors across clusters defined by *region*.

Linear regression						
(Std. Err. adjusted for 5 clusters in region)						
logco2	Coef.	Robust Std. Err.	t	P> t	[95% Conf. Interval]	
loggdp0	1.116759	.0744462	15.00	0.000	.9100631	1.323454
urban0	.0024787	.0015223	1.63	0.179	-.0017478	.0067052
urb_gdp	-.0082808	.0022161	-3.74	0.020	-.0144336	-.0021279
_cons	.8950411	.0726082	12.33	0.000	.6934485	1.096634

Again, the regression coefficients and R^2 are identical to those in the earlier models, but the standard errors, confidence intervals and hypothesis tests have changed. The clustered standard errors are substantially larger than those in the earlier models, resulting in smaller t statistics and higher probabilities. Using `vce(robust)` earlier brought much less change, indicating no particular problem with assuming that errors are independent and identically distributed with respect to predictors in the model. Using `vce(cluster region)`, however, brought larger changes indicating that, as we suspected, errors are not independent and identically distributed with respect to *region*. Consequently, the `vce(cluster region)` estimates are more plausible, and should be reported in place of the default estimates if we were writing up these results as research.

Predicted Values and Residuals

After any regression, the `predict` command can obtain not only predicted values, but also residuals and other post-estimation *case statistics* — statistics that have separate values for each observation in the data. Switching to a different example for this section, we will look at the simple regression of September Arctic sea ice *area* on *year* (using *Arctic9.dta*). A downward trend averaging $-.076$, or roughly 76,000 km² per year, explains 75% of the variance in *area* over these years (1979–2011).

. use C:\data\Arctic9.dta, clear				
. describe year area tempN				
variable	name	storage	display	value
			format	label
year		int	%ty	Year
area		float	%9.0g	Sea ice area, million km ²
tempN		float	%9.0g	Annual air temp anomaly 64N-90N C

```
. regress area year
```

Source	SS	df	MS	Number of obs = 33
Model	17.4995305	1	17.4995305	F(1, 31) = 99.55
Residual	5.4491664	31	.175779561	Prob > F = 0.0000
Total	22.9486969	32	.717146777	R-squared = 0.7626
				Adj R-squared = 0.7549
				Root MSE = .41926

area	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
year	-.0764773	.0076648	-9.98	0.000	-.0921098 -.0608447
_cons	157.4225	15.29154	10.29	0.000	126.2352 188.6098

We can create a new variable named *areahat*, containing predicted values from this regression, and *areares*, containing residuals, through the post-regression command **predict**. Predicted values have the same mean as the original *y* variable. Residuals have a mean of zero ($-1.38e-09 = -1.38 \times 10^{-9} \approx 0$). Note use of a wild-card symbol in the **summarize** command: *area** means all variable names that begin with “area”.

```
. predict areahat
. label variable areahat "Area predicted from year"
. predict areares, resid
. label variable areares "Residuals, area predicted from year"
. summarize area*
```

Variable	Obs	Mean	Std. Dev.	Min	Max
area	33	4.850303	.8468452	3.09	6.02
areahat	33	4.850303	.7395001	3.626667	6.073939
areares	33	-1.38e-09	.4126578	-.8425758	1.116174

The predicted values and residuals can be analyzed like any other variables. For example, we can test residuals for normality to check on the normal-errors assumption. In this example, a skewness-kurtosis test (**sktest**) shows that the residuals distribution is not significantly different from normal ($p = .45$).

```
. sktest areares
```

Variable	Obs	Skewness/Kurtosis tests for Normality			
		Pr(Skewness)	Pr(Kurtosis)	adj chi2(2)	Prob>chi2
areares	33	0.2951	0.5344	1.59	0.4520

Graphing predicted values against *year* traces the regression line (Figure 7.9).

```
. graph twoway connect area areahat year, msymbol(0 +)
```

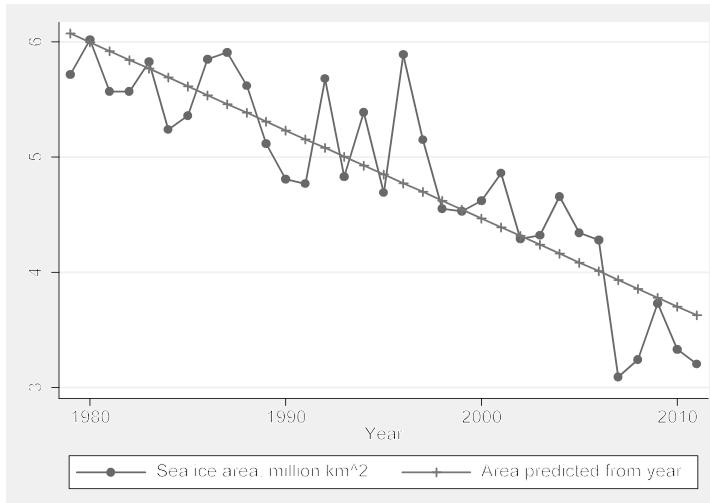


Figure 7.9

Residuals contain information about where the model fits poorly, which is helpful for diagnostic or troubleshooting analysis. Such analysis might begin just by sorting and examining the residuals. Negative residuals occur when our model overpredicts the observed values. That is, in certain years the ice area is lower than we might expect based on the overall trend. To list the years with the five lowest residuals, type

```
. sort areares
. list year area areahat areares in 1/5
```

	year	area	areahat	areares
1.	2007	3.09	3.932576	-.8425758
2.	2008	3.24	3.856098	-.6160985
3.	1984	5.24	5.691553	-.4515528
4.	2011	3.2	3.626667	-.4266666
5.	1990	4.81	5.232689	-.4226894

Three of the five lowest residuals occurred in the most recent five years, indicating that our trend line overestimates recent ice area.

Positive residuals occur when actual y values are higher than predicted. Because the data already have been sorted by e , to list the five highest residuals we add the qualifier `in -5/1`. “ -5 ” in this qualifier means the 5th-from-last observation, and the letter “el” (note that this is not the number “1”) stands for the last observations. The qualifiers `in -5/-1`, `in 47/1` or `in 47/51` each could accomplish the same thing.

```
. list year area areahat areares in -5/1
```

	year	area	areahat	areares
29.	1994	5.39	4.92678	.4632196
30.	2001	4.86	4.391439	.4685608
31.	2004	4.66	4.162007	.4979923
32.	1992	5.68	5.079735	.600265
33.	1996	5.89	4.773826	1.116174

Again there is a pattern: the highest positive residuals, or years when a linear model predicts less ice than observed, occur in the 1990s through early 2000s. Before going on to other analysis, we should re-sort the data by typing `sort year`, so they again form an orderly time series.

Graphs of residuals against predicted values, often called residual-versus-fitted or e-versus-yhat plots, provide a useful diagnostic tool. Figure 7.10 shows a such a graph of `areares` versus `areahat`, with a horizontal line drawn at 0, the residual mean. Later in this chapter, Figure 7.17 shows another way to draw such plots.

```
. graph twoway scatter areares areahat, yline(0)
```

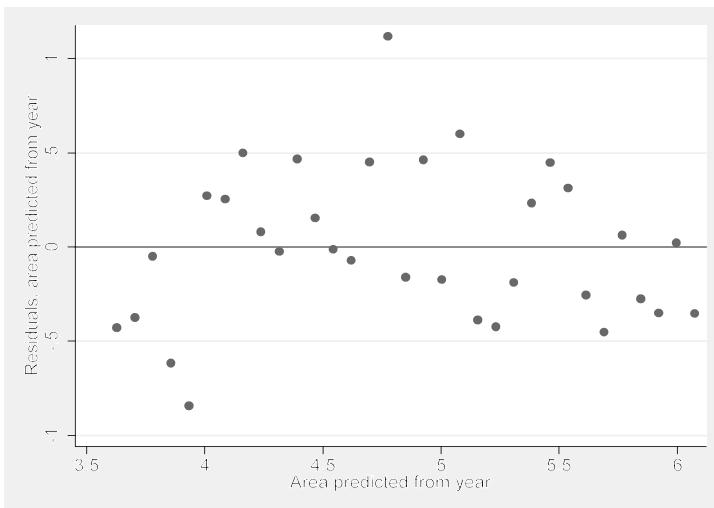


Figure 7.10

Ideally, residual versus predicted graphs show no pattern at all, like a swarm of bees that is denser in the middle (see texts such as Hamilton 1992a for good and bad examples). There *is* a pattern visible in Figure 7.10, however. Predictions are too high in the early years, resulting in mostly negative residuals; too low in the middle, resulting in mostly positive residuals; then too high toward the end, where residuals again turn negative. This is the same pattern noticed earlier when we sorted the residuals by size.

The up and down pattern of residuals becomes obvious in Figure 7.11 in which the residual versus predicted graph is overlaid by a lowess curve. Lowess (locally weighted scatterplot

smoothing) regression provides a broadly useful method for detecting patterns in noisy data. It was briefly introduced in Chapter 3 (Figure 3.26), and will be explained in Chapter 8.

```
. graph twoway scatter areares areahat
    || lowess areares areahat || , yline(0)
```

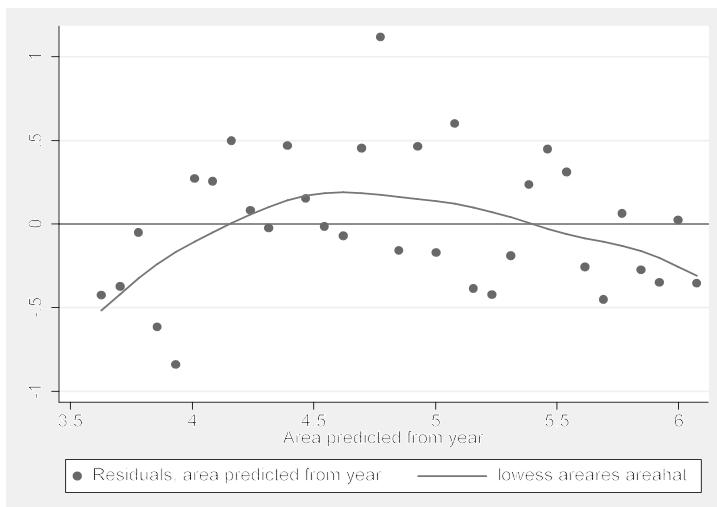


Figure 7.11

This nonlinear pattern in residuals suggests that our linear model was not appropriate for these data. We return to this point in the next section, and again in Chapter 8.

After any regression, Stata temporarily remembers coefficients and other details. Thus `_b[varname]` refers to the coefficient on independent variable `varname`. `_b[_cons]` refers to the coefficient on `_cons` (usually, the *y*-intercept).

```
. display _b[year]
-.07647727

. display _b[_cons]
157.42247
```

Although `predict` makes it easy to calculate predicted values and residuals, we could have defined the same variables through a pair of `generate` statements using the `_b[]` coefficients. The resulting variables, named `areahat1` and `areares1` below, have exactly the same properties as the predicted values and residuals from `predict`, but for certain purposes the `generate` approach gives users more flexibility.

```
. generate areahat1 = _b[_cons] + _b[year]*year
. generate areares1 = area - areahat1
. summarize area*
```

Variable	Obs	Mean	Std. Dev.	Min	Max
area	33	4.850303	.8468452	3.09	6.02
areahat	33	4.850303	.7395001	3.626667	6.073939
areares	33	-1.38e-09	.4126578	-.8425758	1.116174
areahat1	33	4.850303	.7395001	3.626667	6.073939
areares1	33	7.22e-09	.4126578	-.8425758	1.116174

Other Case Statistics

predict can also calculate many other case statistics appropriate for the recently-fitted model. After **regress** (or **anova**), **predict** options include the following. Substitute any new variable name for *new* in these examples.

- . **predict new** Predicted values of *y*. **predict new, xb** means the same thing (referring to **Xb**, the vector of predicted *y* values).
- . **predict new, resid** Residuals.
- . **predict new, rstandard** Standardized residuals.
- . **predict new, rstudent** Studentized (jackknifed) residuals, measuring the *i*th observation's influence on the *y*-intercept.
- . **predict new, stdp** Standard errors of predicted mean *y*.
- . **predict new, stdf** Standard errors of predicted individual *y*, sometimes called the standard errors of forecast or the standard errors of prediction.
- . **predict new, hat** Diagonal elements of hat matrix (**leverage** also works).
- . **predict new, cooksd** Cook's *D* influence, measuring the *i*th observation's influence on all coefficients in the model (or, equivalently, on all *n* predicted *y* values).

Further options obtain predicted probabilities and expected values; type **help regress** for a list.

For illustration, we return to the trend in Arctic sea ice (*Arctic9.dta*). Residual analysis in the previous section suggests that a linear model, Figure 7.9, is not appropriate for these data. One simple alternative termed *quadratic regression* involves regressing the dependent variable on both *year* and *year* squared. We could start by generating a new variable equal to *year* squared:

```
. generate year2 = year^2
. regress area year year2
```

Source	SS	df	MS	Number of obs = 33
Model	18.7878137	2	9.39390686	F(2, 30) = 67.73
Residual	4.16088316	30	.138696105	Prob > F = 0.0000
Total	22.9486969	32	.717146777	R-squared = 0.8187
				Adj R-squared = 0.8066
				Root MSE = .37242

area	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
year	9.658356	3.194155	3.02	0.005	3.135022 16.18169
year2	-.0024398	.0008005	-3.05	0.005	-.0040747 -.0008049
_cons	-9552.853	3186.119	-3.00	0.005	-16059.78 -3045.931

This quadratic regression displays an overall improvement in fit: $R^2 = .8066$, compared with $.7549$ for the linear regression. Moreover, the improvement is statistically significant, as evidenced by the significant coefficient on *year2* ($p = .005$). A scatterplot (Figure 7.12) visualizes this improvement: the quadratic model does not systematically over-then under-then over-predict the actual ice area, as happened with our previous linear model.

```
. predict areahat2
. label variable areahat2 "quadratic prediction"
. graph twoway connect area areahat2 year, msymbol(Oh d)
```

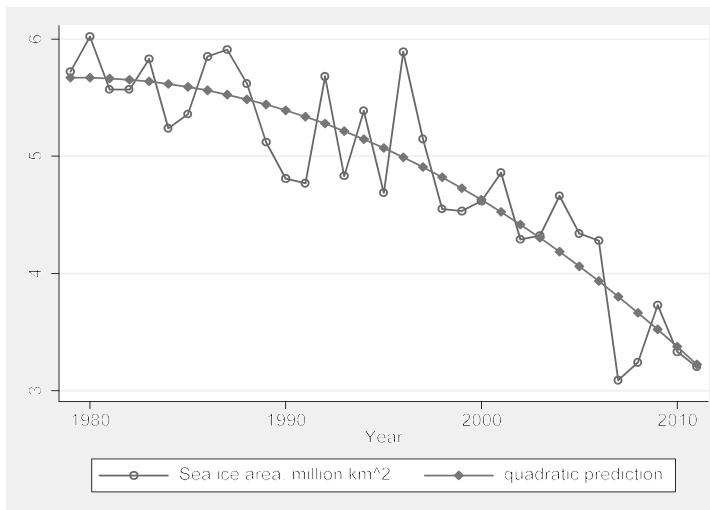


Figure 7.12

Two alternative ways to conduct the same quadratic regression, without first generating a new variable for *year* squared, would be to use Stata's interaction (#) notation. All three of the following commands estimate the same model. Either of the # versions would be needed for a subsequent **margins** command to work properly.

```
. regress area year year2
. regress area year c.year#c.year
. regress area c.year##c.year
```

Although improvement visible in Figure 7.12 is encouraging, quadratic regression can introduce or worsen some statistical problems. One such problem is leverage, meaning the potential influence of observations with unusual x values, or unusual combinations of x values. Leverage statistics, also called hat matrix diagonals, can be obtained by **predict**. In this example we unimaginatively name the leverage measure *leverage*.

```
. predict leverage, hat
```

Figure 7.13 draws the quadratic regression curve again, this time making the area of **connect** marker symbols proportional to *leverage* by specifying analytical weights, [**aw=leverage**]. This graph also overlays a median spline plot (**mspline**) which draws a smooth curve connecting the predicted values *areahat2*. Median spline plots often serve better than line or connected-line plots to graph a curvilinear relationship. For this purpose it helps to specify a large number of **bands()**; type **help twoway mspline** for more about this **twoway** plot type.

```
. graph twoway connect area year [aw=leverage], msymbol(Oh)
    || mspline areahat2 year, bands(50) lwidth(medthick)
    || ,note("Data points area proportional to leverage")
    legend(off) xline(1980 1996) xlabel(1980(5)2010, grid) xtitle("")
    ytitle("Sea ice area, million km{superscript:2}")
```

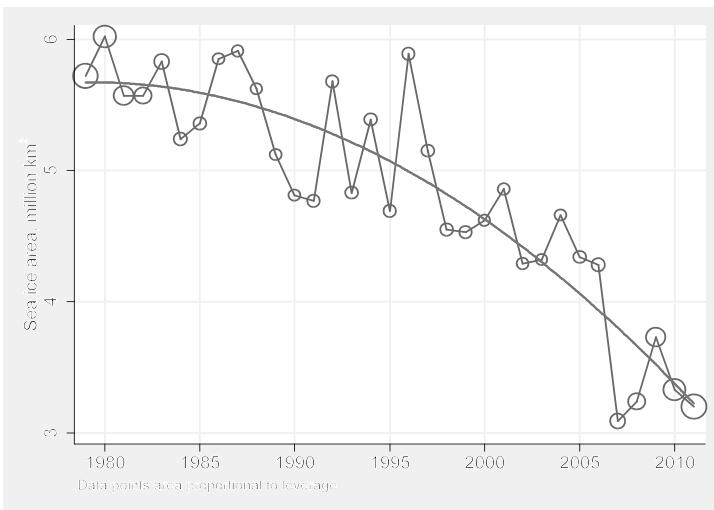


Figure 7.13

The marker symbols in Figure 7.13 are largest for the first and last years, because these have the most leverage. Quadratic regression tends to exaggerate the importance of extreme x values (which, through squaring, become even more extreme) so the model tracks those values closely. Note how well the curve fits the first and last years.

Leverage reflects the potential for influence. Other statistics measure influence directly. One type of influence statistic, DFBETAS, indicates by how many standard errors the coefficient on x_1 would change if observation i were dropped from the regression. These can be obtained for

a single predictor through a **predict new, dfbeta(xvarname)** command. DFBETAS for all of the predictors in a model are obtained more easily by the **dfbeta** command, which creates a new variable for each predictor. In our example there are two predictors, *year* and *year2*. Consequently the **dfbeta** command defines two new influence statistics, one for each predictor.

```
. dfbeta
. describe _dfbeta*
      storage   display      value
variable name   type    format   label
                               variable label
_dfbeta_1        float   %9.0g      dfbeta year
_dfbeta_2        float   %9.0g      dfbeta year2
. summarize _dfbeta*
      variable   obs      mean     std. dev.      min      max
_dfbeta_1          33  .0007181  .1702339 -.3360457  .550404
_dfbeta_2          33 - .0007234  .1702921 -.550294  .3353676
```

Box plots in Figure 7.14 show the distributions of both DFBETAS variables, with outliers labeled by *year*.

```
. graph box _dfbeta*, marker(1, mlabel(year)) marker(2, mlabel(year))
```

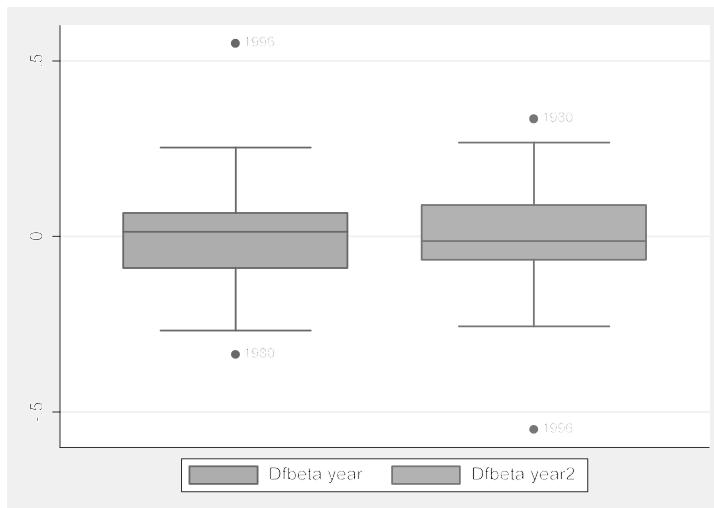


Figure 7.14

The *_dfbeta_1* values measure the influence of each observation on the coefficient for *year*. 1996 (visible as a particularly high value in Figure 7.13) exerts the most influence on the *year* coefficient. The *_dfbeta_1* = .55 tells us that the coefficient on *year* in the full-sample regression is *about .55 standard errors higher than it would be, if we re-estimated the model with 1996 set aside*. Similarly, the negative value for 1980, *_dfbeta_1* = -.336, indicates that the full-sample coefficient on *year* is *about .336 standard errors lower than it would be, if we re-estimated the model with 1980 set aside*. And while 1996 makes the coefficient on *year* higher than it would

otherwise be, it makes the coefficient on *year2* lower by a similar amount. The reverse applies to 1980. In the original data (Figure 7.13) 1980 is not much of an outlier, but it has relatively high leverage which makes it more influential than other equally outlying but lower-leverage years.

If we were unsure about how to read DFBETAS, we could confirm our interpretation by repeating the regression with influential observations set aside. To save space, **quietly** here suppresses the regression output, and we ask only to display the coefficients on *year* and *year2*, separated by two blank spaces for readability. The first regression employs the full sample. The second sets aside 1996, which makes the coefficient on *year* lower (from 9.658 to 8.067) and the coefficient on *year2* higher (from -.0024 to -.0020). Setting aside 1980 in the third regression has an opposite effect: the coefficient on *year* becomes higher (9.658 to 10.730) and the coefficient on *year2* becomes lower (-.0024 to -.0027).

```
. quietly regress area year year2
. display _b[year] " " _b[year2]
9.6583565 -.00243981

. quietly regress area year year2 if year!=1996
. display _b[year] " " _b[year2]
8.0666424 -.00204096

. quietly regress area year year2 if year!=1980
. display _b[year] " " _b[year2]
10.730059 -.00270786
```

If we use # to enter the squared term instead, the last commands would be as follows.

```
. quietly regress area year c.year#c.year if year!=1980
. display _b[year] " " _b[c.year#c.year]
10.730059 -.00270786
```

Leverage, DFBETAS or any other case statistic could be used directly to exclude observations from an analysis. For example, the first two commands below calculate Cook's *D*, a statistic that measures influence of each observation on the model as a whole, instead of on individual coefficients as DFBETAS do. The third command repeats the initial regression using only those observations with Cook's *D* values below .10.

```
. regress area year year2
. predict D, cooksd
. regress area year year2 if D <.10
```

Using any fixed definition of what constitutes an outlier, we are liable to see more of them in larger samples. For this reason, sample-size-adjusted cutoffs are sometimes recommended for identifying unusual observations. After fitting a regression model with *K* coefficients (including the constant) based on *n* observations, we might look more closely at those observations for which any of the following are true:

leverage $h > 2K/n$

Cook's $D > 4/n$

$DFITS > 2\sqrt{K/n}$

Welsch's $W > 3\sqrt{K}$

$$\begin{aligned} DFBETA &> 2/\sqrt{n} \\ |COVRATIO - 1| &\geq 3K/n \end{aligned}$$

The reasoning behind these cutoffs, and the diagnostic statistics more generally, can be found in Cook and Weisberg (1982, 1994), Belsley, Kuh and Welsch (1980) or Fox (1991).

Diagnosing Multicollinearity and Heteroskedasticity

Multicollinearity refers to the problem of too-strong linear relationships among the predictors or independent variables in a model. If perfect multicollinearity exists among the predictors, regression equations lack unique solutions. Stata warns us and then drops one of the offending predictors. High but not perfect multicollinearity causes more subtle problems. When we add a new predictor that is strongly related to predictors already in the model, symptoms of possible trouble include the following:

- Substantially higher standard errors, with correspondingly lower t statistics.
- Unexpected changes in coefficient magnitudes or signs.
- Nonsignificant coefficients despite a high R^2 .

Multiple regression attempts to estimate the independent effects of each x variable. There is little information for doing so, however, if one or more of the x variables does not have much independent variation. The symptoms listed above warn that coefficient estimates have become unreliable, and might shift drastically with small changes in the data or model. Further troubleshooting is needed to determine whether multicollinearity really is at fault and, if so, what should be done about it.

Correlation matrices often are consulted as a diagnostic, but they have limited value for detecting multicollinearity. Correlations can only reveal *collinearity*, or linear relationships among variable pairs. *Multicollinearity*, on the other hand, involves linear relationships among any combination of independent variables, which might not be apparent from correlations. Better diagnostics are provided by the postestimation command **estat vif**, for variance inflation factor.

. regress area year year2

Source	SS	df	MS	Number of obs = 33
Model	18.7878137	2	9.39390686	F(2, 30) = 67.73
Residual	4.16088316	30	.138696105	Prob > F = 0.0000
Total	22.9486969	32	.717146777	R-squared = 0.8187
				Adj R-squared = 0.8066
				Root MSE = .37242

area	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
year	9.658356	3.194155	3.02	0.005	3.135022 16.18169
year2	-.0024398	.0008005	-3.05	0.005	-.0040747 -.0008049
_cons	-9552.853	3186.119	-3.00	0.005	-16059.78 -3045.931

. estat vif

Variable	VIF	1/VIF
year	220094.55	0.000005
year2	220094.55	0.000005
Mean VIF	220094.55	

The 1/VIF column at right in an **estat vif** table gives values equal to $1 - R^2$ from the regression of each x on the other x variables. Our Arctic ice example presents an extreme case: only .000005, less than .0005 percent, of the variance of *year* is independent of *year2* (or vice versa). The variance inflation factor or VIF values themselves indicate that, with both predictors in the model, the variance of their coefficients is about 220,000 times higher than it otherwise would be. The standard error on *year* in the simple linear model seen earlier in this chapter is .0076648; the corresponding standard error in the quadratic model above is many times higher, 3.194155.

These VIF values indicate serious collinearity. Because there are only two predictors, in this instance we can confirm that observation just by looking at their correlations. *year* and *year2* correlate almost perfectly.

. correlate area year year2
(obs=33)

	area	year	year2
area	1.0000		
year	-0.8732	1.0000	
year2	-0.8737	1.0000	1.0000

Collinearity or multicollinearity can occur in any kind of model, but they are particularly common in models where some predictors are defined from others — such as those with interaction effects, or quadratic regression. The simple trick of centering, recommended earlier for interaction terms, can help with quadratic regression too. By reducing multicollinearity, centering often yields more precise coefficient estimates with lower standard errors.

We accomplish centering by subtracting the mean from one x variable before calculating the second. The mean *year* in these data is 1995; a centered version here called *year0* represents years before 1995 (negative) or after 1995 (positive). Centered *year0* has a mean of 0. A second new variable, *year02*, equals *year0* squared. Its values range from 256 (when *year0* = -16, meaning 1979) to 0 (when *year0* = 0, meaning 1995), and then back up to 256 (when *year0* = +16, meaning 2011).

. gen year0 = year - 1995
. gen year02 = year0 ^2
. summarize year year0 year02

Variable	Obs	Mean	Std. Dev.	Min	Max
year	33	1995	9.66954	1979	2011
year0	33	0	9.66954	-16	16
year02	33	90.66667	82.23847	0	256

After centering, differences will be most noticeable in the centered variable's coefficient and standard error. To see this, we regress sea ice *area* on *year0* and *year02*. The R^2 and overall F test are exactly as they were in the regression of *area* on uncentered *year* and *year2*. Predicted

values and residuals will be the same too. In the centered version, however, standard errors on *year0* are much lower, confidence intervals narrower, *t* statistics larger, and *t* test results “more significant” than they were in the uncentered regression with plain *year*.

```
. regress area year0 year02
```

Source	SS	df	MS	Number of obs = 33 F(2, 30) = 67.73 Prob > F = 0.0000 R-squared = 0.8187 Adj R-squared = 0.8066 Root MSE = .37242			
Model	18.7878137	2	9.39390686				
Residual	4.16088316	30	.138696105				
Total	22.9486969	32	.717146777				
area	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]		
year0	-.0764773	.0068085	-11.23	0.000	-.0903821	-.0625725	
year02	-.0024398	.0008005	-3.05	0.005	-.0040747	-.0008049	
_cons	5.071512	.0973195	52.11	0.000	4.872759	5.270265	

Although *year* and *year2* correlate almost perfectly, centered versions have a correlation near 0.

```
. correlate year0 year02  
(obs=33)
```

	year0	year02
year0	1.0000	
year02	-0.0000	1.0000

Because the two predictors are uncorrelated, no variance inflation occurs.

```
. estat vif
```

Variable	VIF	1/VIF
year0	1.00	1.000000
year02	1.00	1.000000
Mean VIF		1.00

The **estat** command obtains other useful diagnostic statistics as well. For example, **estat hettest** performs a heteroskedasticity test for the assumption of constant error variance. It does this by examining whether squared standardized residuals are linearly related to predicted values (see Cook and Weisberg 1994 for discussion and example). As seen below, **estat hettest** here gives no reason to reject the null hypothesis of constant variance. That is, we see no significant heteroskedasticity.

```
. estat hettest
```

```
Breusch-Pagan / Cook-Weisberg test for heteroskedasticity
Ho: Constant variance
Variables: fitted values of area

chi2(1)      =    0.00
Prob > chi2  =   0.9802
```

Presence of significant heteroskedasticity, on the other hand, would imply that our standard errors could be biased, and the resulting hypothesis tests invalid.

Confidence Bands in Simple Regression

This section introduces some additional graphics that help to visualize a regression model or diagnose possible problems. Continuing with the *Arctic9.dta* data, variable *tempN* describes mean annual air temperature anomalies for the entire region from 64 to 90 degrees north latitude, estimated from land and sea surface records by NASA. Temperature anomalies represent differences, in degrees Celsius, from regional temperatures over the reference years 1951–1980. Positive anomalies thus represent above-average temperatures, relative to 1951–1980.

We have seen that area, extent and volume of Arctic sea ice (especially at their September minimum) have declined over the 1979–2011 period of satellite observation. Unsurprisingly, Arctic surface air temperatures warmed over this same period, although this is not the whole cause for sea ice decline. The warming trend amounts to about .058 °C/year, or .58 °C/decade — considerably faster than the globe as a whole. For comparison, other NASA data not shown here indicate a global warming trend of just .16 °C/decade over these years.

```
. regress tempN year
```

Source	SS	df	MS	Number of obs = 33
Model	10.2449886	1	10.2449886	F(1, 31) = 51.64
Residual	6.15050844	31	.198403498	Prob > F = 0.0000
Total	16.395497	32	.512359282	R-squared = 0.6249
				Adj R-squared = 0.6128
				Root MSE = .44543
tempN	Coef.	Std. Err.	t	P> t [95% Conf. Interval]
year	.058516	.0081432	7.19	0.000 .0419079 .0751242
_cons	-115.9492	16.24582	-7.14	0.000 -149.0828 -82.81563

Figure 7.15 plots this upward trend in Arctic temperature anomalies. Actual temperatures are overlaid on regression line with 95% confidence interval for the conditional mean, as specified by the **twoway lfitci, stdp** part of this command. Other options call for a medium-thick regression line, and medium-large text for the main title. A degree symbol, ASCII character 186, is inserted in the *y*-axis title (see Figure 3.16 in Chapter 3 for other ASCII characters).

```
. graph twoway lfitci tempN year, stdp lwidth(medthick)
|| connect tempN year, msymbol(Th)
|| , ytitle("Annual temperature anomaly, `=char(186)'C")
legend(off) xlabel(1980(5)2010) yline(0)
title("Arctic temperature trend with 95% c.i. for
conditional mean", size(medlarge))
```

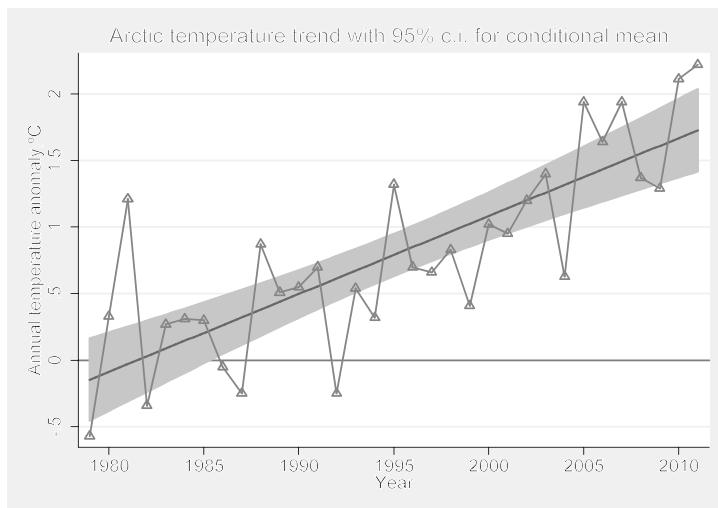


Figure 7.15

Many yearly values lie outside the confidence intervals in Figure 7.15, emphasizing the fact that these intervals refer to conditional mean values, or the trend itself, rather than to individual predictions. Suppose we wished to make an individual prediction for the year 2012, and also find an appropriate confidence interval for this prediction. One way to do that would be to use the Data Editor to add a new 34th row of data, containing only the *year* value 2012. Alternatively, the following two commands accomplish the same thing.

```
. set obs 34
. replace year = 2012 in 34
```

Then repeat the regression to obtain predicted values and standard errors of forecasts (**stdf**). Upper and lower 95% confidence limits are approximately the predicted values minus or plus twice the standard error of forecasts: tempNhat minus or plus $2 \times \text{tempNse}$.

```
. predict tempNhat
. label variable tempNhat "Predicted temperature"
. predict tempNse, stdf
. label variable tempNse "Standard error of forecast"
. gen tempNlo = tempNhat - 2*tempNse
. label variable tempNlo "lower confidence limit"
. gen tempNhi = tempNhat + 2*tempNse
. label variable tempNhi "upper confidence limit"
. list year tempN* in -5/1
```

	year	tempN	tempNhat	tempNse	tempNlo	tempNhi
30.	2008	1.37	1.551012	.4643515	.6223085	2.479715
31.	2009	1.29	1.609528	.4662754	.6769768	2.542078
32.	2010	2.11	1.668044	.468333	.7313777	2.60471
33.	2011	2.22	1.72656	.4705225	.7855148	2.667605
34.	2012	.	1.785076	.4728422	.8393915	2.73076

We might now graph the *tempNlo* and *tempNhi* values in a range area (**twoway rarea**), range spike (**rspike**), capped spike (**rcap**) or similar plot to show the confidence intervals. Figure 7.16 takes a simpler approach using **twoway lfitci, stdf range(1979 2012)**. We overlay both a connected-line (**connect**) plot of observed temperatures 1979–2011 with hollow triangles as markers (**msymbol(Th)**), and a scatterplot of predicted temperature for 2012 only, with a square marker symbol (**msymbol(S)**). Added text states the numerical predicted value and confidence limits (*tempNhat*, *tempNlo* and *tempNhi* for 2012, copied from the table above). Note that the **lfitci, stdf** confidence band is wide enough to cover roughly 95% of the observations, unlike the **lfit, stdp** band in Figure 7.15.

```
. graph twoway lfitci tempN year, stdf lwidth(medthick)
    range(1979 2012)
    || connect tempN year, msymbol(Th)
    || scatter tempNhat year if year==2012, msymbol(S)
    || , ytitle("Annual temperature anomaly, `=char(186)'C")
    legend(off) xlabel(1980(5)2010) yline(0)
    text(2.8 2007 "2012 predicted temp"
        "1.79 (.84-2.73) `=char(186)'C")
    title("Arctic temperature trend with 95% c.i. for predictions"
        , size(medlarge))
```

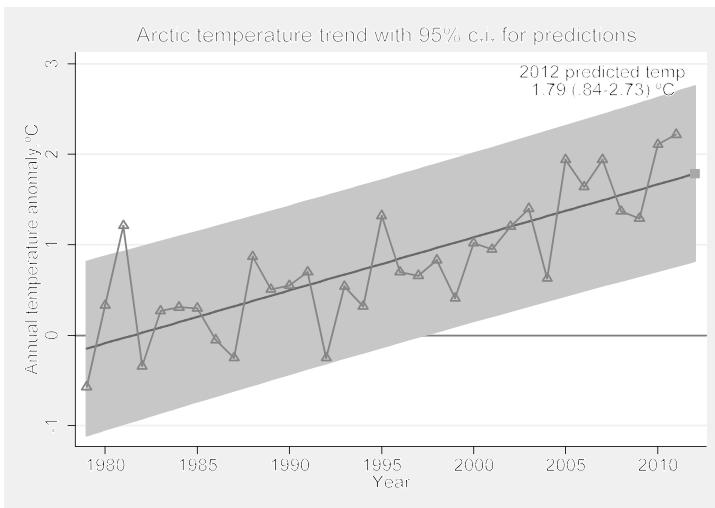


Figure 7.16

The temperature, ice area and other variables in *Arctic9.dta* form time series, a type of data that often exhibits autocorrelation, or serial correlation between successive data values. If regression errors are in fact autocorrelated, then the usual formulas for standard errors, confidence intervals and hypothesis tests — such as those used in this section — could be misleading. Consequently, researchers modeling time series data routinely check for residual autocorrelation, and apply specialized time series regression methods when it is present.

Time series regression methods (Chapter 12) require data that are declared as time series using the **tset** command. This identifies the variable providing an index of time.

```
. tset year
      time variable: year, 1979 to 2011
                  delta: 1 year
```

For **tset** data, several methods become available to check for autocorrelation. One well known but minimally informative method is the Durbin–Watson test.

```
. estat dwatson
durbin-watson d-statistic( 2,     33) =  2.091689
```

Textbooks often contain look-up tables for the Durbin–Watson test. With 33 observations and 2 parameters estimated, the calculated value of 2.09 lies well above the $\alpha = .05$ table's upper limit (1.51), so we do not reject the null hypothesis that there is no first-order, positive autocorrelation. That is good news for the validity of Figures 7.15–16. It offers no assurance about autocorrelation at other lags such as two, three or four years previously, however.

A more informative approach calculates autocorrelation coefficients for the residuals at many lags, with a cumulative portmanteau test or Ljung–Box *Q* statistic. This test is accomplished by applying **corrgram** to the model residuals (here named *tempNres*).

```
. predict tempNres, resid
. corrgram tempNres
```

LAG	AC	PAC	Q	Prob>Q	-1 [Autocorrelation]	0 [Partial Autocor]	1 [Autocor]
1	-0.0803	-0.0826	.23248	0.6297			
2	-0.0920	-0.1081	.54749	0.7605			
3	-0.0494	-0.0746	.64146	0.8869			
4	-0.0249	-0.0461	.66619	0.9554			
5	0.2623	0.2818	3.5048	0.6227			
6	-0.1982	-0.2320	5.1858	0.5202			
7	0.1972	0.2678	6.9135	0.4379			
8	-0.0025	0.0217	6.9138	0.5460			
9	-0.1696	-0.2945	8.2974	0.5045			
10	0.1652	0.3323	9.6677	0.4701			
11	-0.2572	-0.5436	13.14	0.2843			
12	0.0647	-0.0919	13.37	0.3428			
13	-0.2205	-0.4464	16.177	0.2397			
14	0.0219	-0.0385	16.206	0.3009			

The *Q* tests in this output do not approach statistical significance at any lags from 1 to 14 years. Thus, **corrgram** more persuasively agrees with **estat dwatson** in finding no significant autocorrelation among residuals from the temperature model. The tests and confidence intervals in this section are not called into question.

Diagnostic Graphs

Stata offers many specialized graphs useful for diagnostic purposes following a regression. A few of these are illustrated in this section; type **help regress postestimation** for a list. Our example here will be an elaboration of the Arctic ice model, in which September sea ice area is predicted from *year* and *year*² (after *year* is centered) along with annual surface air temperature anomaly (*tempN*). Centered *year* (*year0*) and its square (*year02*) were calculated earlier, but are generated again here assuming they no longer exist in memory. The three predictors together explain about 82% of the variance in ice area.

```
. use C:\data\Arctic9.dta, clear
. gen year0 = year -1995
. gen year02 = year0 ^2
. regress area year0 year02 tempN
```

Source	SS	df	MS	Number of obs = 33			
Model	19.2134599	3	6.40448663	F(3, 29) = 49.72			
Residual	3.735237	29	.128801276	Prob > F = 0.0000			
Total	22.9486969	32	.717146777	R-squared = 0.8372			
				Adj R-squared = 0.8204			
				Root MSE = .35889			

area	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
year0	-.0601115	.0111399	-5.40	0.000	-.0828951 -.0373279
year02	-.0019336	.0008202	-2.36	0.025	-.0036111 -.0002562
tempN	-.2796799	.1538498	-1.82	0.079	-.594338 .0349783
_cons	5.24665	.1344514	39.02	0.000	4.971666 5.521634

As in the previous section, a test for residual autocorrelation is prudent. The *Q* test finds no significant autocorrelation at lags 1 through 10 — that is, comparing residuals from each year with residuals from 1 through 10 years previously. Some autocorrelation does appear at lags longer than 10, but that is unlikely to affect our results.

```
. predict areares2, resid
. corrgram areares2, lag(10)
```

LAG	AC	PAC	Q	Prob>Q	-1 [Autocorrelation]		0	1 -1 [Partial Autocor]		0	1
1	0.1140	0.1141	.46917	0.4934							
2	-0.1826	-0.2003	1.7112	0.4250							
3	-0.3273	-0.2968	5.8358	0.1199							
4	-0.0554	-0.0157	5.9581	0.2023							
5	0.0238	-0.1040	5.9816	0.3080							
6	-0.1620	-0.4049	7.1046	0.3113							
7	-0.1077	-0.1646	7.62	0.3673							
8	0.2332	0.3384	10.132	0.2559							
9	0.3583	0.2410	16.309	0.0607							
10	-0.0160	-0.2435	16.322	0.0908							

A residual-versus-fitted plot could be drawn by calculated predicted values and graphing *areares2* against those. A faster way is by the **rvfplot** command. The example in Figure 7.17 adds a horizontal line marking zero, the residual mean. It also labels the data points by *year*. The plot reveals one outlier with a high positive residual (1996) but no obvious signs of trouble.

```
. rvfplot, yline(0) mlabel(year)
```

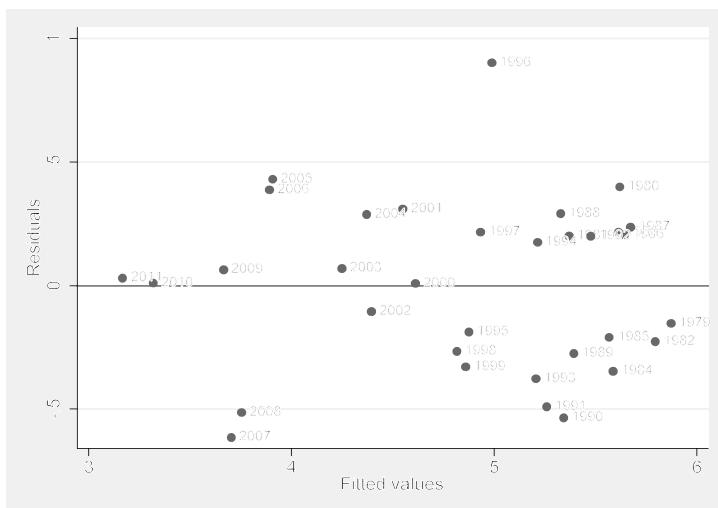


Figure 7.17

Added-variable plots are valuable diagnostic tools known by different names including partial-regression leverage plots, adjusted partial residual plots, or adjusted variable plots. They depict the relationship between y and one x variable, adjusting for the effects of other x variables. If we regressed y on x_2 and x_3 , and likewise regressed x_1 on x_2 and x_3 , then took the residuals from each regression and graphed these residuals in a scatterplot, we would obtain an added-variable plot for the relationship between y and x_1 , adjusted for x_2 and x_3 . An **avplot** command performs the necessary calculations automatically. We can draw the added-variable plot for predictor $tempN$, for example, just by typing

```
. avplot tempN
```

Speeding the process further, we could type **avplots** to obtain a complete set of tiny added-variable plots with each of the predictor variables in the preceding regression. Figure 7.18 shows the results from the regression of *area* on *year0*, *year02* and *tempN*. The lines drawn in added-variable plots have slopes equal to the corresponding partial regression coefficients. For example, the slope of the line at lower left in Figure 7.18 equals $-.2797$, exactly the coefficient on *tempN* in our 3-predictor regression.

```
. avplots
```

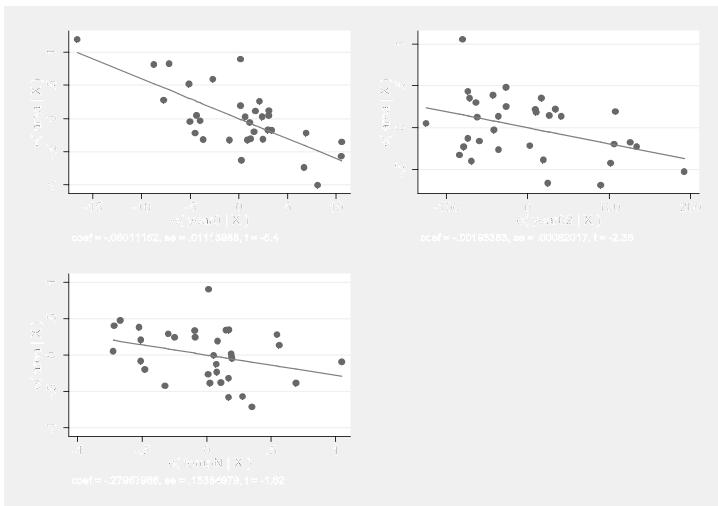


Figure 7.18

Added-variable plots help to identify observations exerting a disproportionate influence on the regression model. In simple regression with one x variable, ordinary scatterplots suffice for this purpose. In multiple regression, however, the signs of influence become more subtle. An observation with an unusual combination of values on several x variables might have high leverage, or potential to influence the regression, even though none of its individual x values is unusual by itself. High-leverage observations show up in added-variable plots as points horizontally distant from the rest of the data. Most of the horizontally extreme points in Figure 7.18 appear at positions consistent with the rest of the data, however.

One high outlier appears in the upper right plot of Figure 7.18, suggesting a possible influence that steepens (makes more negative) the coefficient on `year02`. When we draw one added-variable plot using the singular `avplot` command, and label its data points, 1996 shows up as the outlier.

```
. avplot year02, mlabel(year)
```

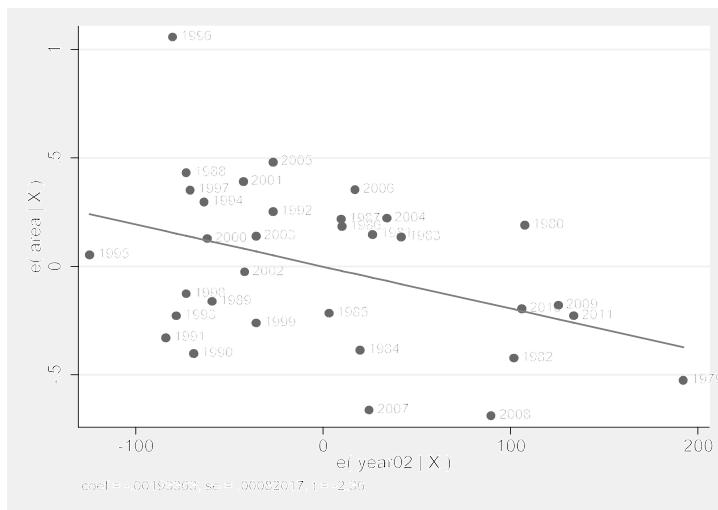


Figure 7.19

Component-plus-residual plots (produced by **cprplot**) take a different approach. A component-plus residual plot for variable $x1$ graphs each residual plus its component predicted from $x1$:

$$e_i + b_1 x_{1i}$$

against values of $x1$. Such plots might help diagnose nonlinearities and suggest alternative functional forms. An augmented component-plus-residual plot (Mallows 1986) works somewhat better, although both types often seem inconclusive. Figure 7.20 shows an augmented component-plus-residual plot from the regression of *area* on *year0*, *year02* and *tempN*.

```
. acprplot tempN, lowess
```

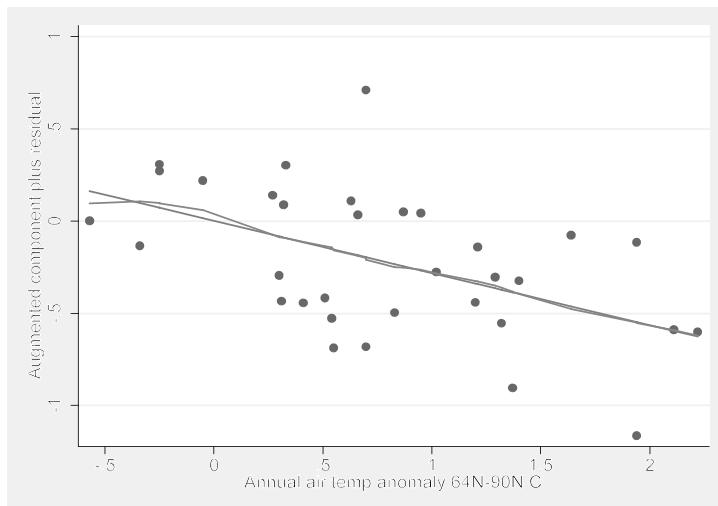


Figure 7.20

The straight line in Figure 7.20 corresponds to the regression model. The curved line reflects lowess smoothing, which would show us if there was much nonlinearity. The curve's downturn at extreme left can be disregarded as a lowess artifact because only a few cases determine its location (see Chapter 8). If more central parts of the lowess curve showed a systematically curved pattern, departing from the linear regression model, we would have reason to doubt the model's adequacy. In Figure 7.20, however, the component-plus-residuals medians closely follow the regression model. This plot reinforces the conclusion that the present regression model adequately accounts for all nonlinearity visible in the raw data, leaving none in its residuals.

As its name implies, a leverage-versus-squared-residuals plot graphs leverage (hat matrix diagonals) against the residuals squared. Figure 7.21 shows such a plot for the *area* regression. To identify individual outliers, we label the markers by *year*. The option **mlabsize(medsmall)** calls for medium small marker labels, somewhat larger than the default size of small. (See **help testsizestyle** for a list of other choices.) **mlabpos(11)** places these labels placed at an 11 o'clock position relative to marker symbols. Most of the years form a jumble at lower left in Figure 7.21, but 1996 once again stands out.

```
. lvr2plot, mlabel(year) mlabsize(medsmall) mlabpos(11)
```

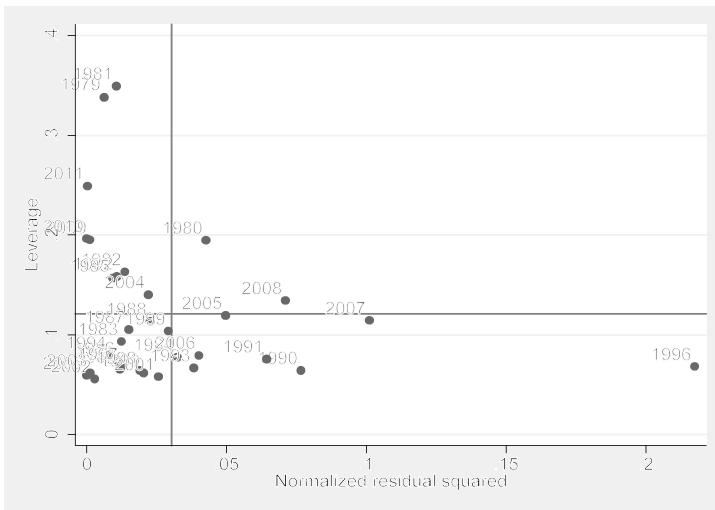


Figure 7.21

Lines in a leverage-versus-squared-residuals plot mark the means of leverage (horizontal line) and squared residuals (vertical line). Leverage tells us how much potential for influencing the regression an observation has, based on its particular combination of x values. Extreme x values or unusual combinations give an observation high leverage. A large squared residual indicates an observation with a y value much different from that predicted by the regression model. 1996 has by far the largest squared residual, indicating that the model fits that year least well. But its combination of *tempN* and *year* values are middle-of-the-road, so 1996 has below-average leverage.

Diagnostic graphs and statistics draw attention to influential or potentially influential observations, but they do not say whether we should set those observations aside. That requires a substantive decision based on careful evaluation of the data and research context. There is no substantive justification for setting aside 1996 in the Arctic ice example, but suppose just for illustration we try that step anyway. As might be expected from 1996's low leverage, omitting this year turns out to make little difference to the *area* regression results. The coefficients on *year0* and *tempN* remain about the same, with *tempN*'s effect now significant. The coefficient on *year02* is a bit closer to zero, but still negative and significant. R^2_a increases slightly, from .82 to .85, with this nonconforming year left out. The fact that these differences are minor, and we have no substantive reason to discard 1996, both argue for keeping it in the analysis.

```
. regress area year0 year02 tempN if year!=1996
```

Source	SS	df	MS	Number of obs =	32
Model	18.9699037	3	6.32330125	F(3, 28) =	61.82
Residual	2.8640433	28	.102287261	Prob > F =	0.0000
Total	21.8339471	31	.704320873	R-squared =	0.8688
				Adj R-squared =	0.8548
				Root MSE =	.31982

area	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
year0	-.0602946	.0099275	-6.07	0.000	-.0806302 -.0399591
year02	-.0015288	.0007439	-2.05	0.049	-.0030527 -4.87e-06
tempN	-.2820721	.1371057	-2.06	0.049	-.5629203 -.0012239
_cons	5.182538	.1218136	42.54	0.000	4.933014 5.432062

Chambers et al. (1983) and Cook and Weisberg (1994) provide more detailed examples and explanations of diagnostic plots and other graphical methods for data analysis.

Advanced Regression Methods

The previous chapter focused on ordinary least squares (OLS) regression. With good reason, OLS is by far the most widely used regression method. This chapter turns to a selection of other methods that also have broad applications, addressing complications that OLS does not. Although computationally more intensive than OLS, based on more challenging mathematics, these advanced regression methods are not much harder to use in Stata.

There is no particular sequence for presenting such diverse topics. Each section of this chapter is written to be self-contained, so readers can browse through for ideas. Examples are kept basic, but with reference to other sources where more detail can be found.

The following menu groups cover most of the operations discussed in this chapter. One topic, nonlinear regression, requires a command-based approach.

Graphics > Twoway graph (scatter, line, etc.)

Statistics > Nonparametric analysis > Lowess smoothing

Statistics > Linear models and related > Other > Robust regression

Statistics > Linear models and related > Quantile regression

Statistics > Linear models and related > Box-Cox regression

Statistics > SEM (Structural Equation Modeling)

Example Commands

. **boxcox y x1 x2 x3, model(lhs)**

Finds maximum-likelihood estimates of the parameter λ (lambda) for a Box–Cox transformation of y , assuming that $y^{(\lambda)}$ is a linear function of $x1$, $x2$ and $x3$ plus Gaussian constant-variance errors. The **model(lhs)** option restricts transformation to the left-hand-side variable y . Other options could transform right-hand-side (x) variables, and control further details of the model. Type **help boxcox** for the syntax and a complete list of options. The *Base Reference Manual* gives technical details.

. **graph twoway mband y x, bands(10) || scatter y x**

Produces a *y* versus *x* scatterplot with line segments connecting the cross-medians (median *x*, median *y* points) within 10 equal-width vertical bands. This is one form of band regression. Typing **mspline** in place of **mband** in this command would result in the cross-medians being connected by a smooth cubic spline curve instead of by line segments.

. **graph twoway lowess y x, bwidth(.4) || scatter y x**

Draws a lowess-smoothed curve with a scatterplot of *y* versus *x*. **bwidth(.4)** calls for lowess calculations using a bandwidth of .4 (40% of the data). In order to generate smoothed values as a new variable, use the related command **lowess** (next example).

. **lowess y x, bwidth(.3) gen(newvar)**

Draws a lowess-smoothed curve on a scatterplot of *y* versus *x*, using a bandwidth of .3 (30% of the data). Predicted values for this curve are saved as a variable named *newvar*. The **lowess** command offers an option to save predicted values, which is not done by **graph twoway lowess**. See **help lowess** for details.

. **n1 (y1 = {b1=1}*{b2=1}^x)**

Uses iterative nonlinear least squares to fit a 2-parameter exponential growth model, $y = b_1 b_2^x$. Two parameters to be estimated, *b1* and *b2*, are bound in {braces}, together with their suggested starting values (1). Instead of writing out our model in the command line, we might save time by calling one of the common models supplied with Stata, or write a new program to define our own model. The 2-parameter exponential happens to be one of these common models, defined by an existing program named **exp2**. Consequently, we could accomplish the same thing as the above command simply by typing

n1 exp2: y x, init(b1 1 b2 1)

After **n1**, use **predict** to generate predicted values or residuals.

. **n1 log4: y x, init(b0 5 b1 25 b2 .1 b3 50)**

Fits a 4-parameter logistic growth model (**log4**) of the form

$$y = b_0 + b_1/(1 + \exp(-b_2(x - b_3)))$$

Sets initial parameter values for the iterative estimation process at *b0* = 5, *b1* = 25, *b2* = .1, and *b3* = 50. **log4**, like **exp2**, is one of the nonlinear models supplied with Stata.

. **sem (y <- x1 x2 x3 x4) (x1 <- x3 x4) (x2 <- x3 x4)**

Structural equation model in which *x1*, *x2*, *x3* and *x4* affect *y*. *x1* and *x2* are intervening variables, each also affected by *x3* and *x4*.

. **rreg y x1 x2 x3**

Performs robust regression of *y* on three predictors, using iteratively reweighted least squares with Huber and biweight functions tuned for 95% Gaussian efficiency. Given appropriately configured data, **rreg** can also obtain robust means, confidence intervals, difference of means tests, and ANOVA or ANCOVA.

. **rreg y x1 x2 x3, nolog tune(6) genwt(rweight) iterate(10)**

Performs robust regression of *y* on three predictors. The options shown above tell Stata not to print the iteration log, to use a tuning constant of 6 (which downweights outliers more steeply than the default 7), to generate a variable (arbitrarily named *rweight*) holding the

final-iteration robust weights for each observation, and to limit the maximum number of iterations to 10.

. qreg y x1 x2 x3

Performs quantile regression, also known as least absolute value (LAV) or minimum $L1$ -norm regression, of y on three predictors. By default, **qreg** models the conditional .5 quantile (approximate median) of y as a linear function of the predictor variables, and thus provides median regression.

. qreg y x1 x2 x3, quantile(.25)

Performs quantile regression modeling the conditional .25 quantile (first quartile) of y as a linear function of $x1$, $x2$ and $x3$.

Lowess Smoothing

Lowess smoothing, already demonstrated without explanation at several points in this book, is a very useful tool for nonparametric regression. Nonparametric regression methods generally do not yield an explicit regression equation, and do not require the analyst to specify a relationship's functional form in advance. Instead, they help to explore the data with an open mind. This process can uncover unexpected and interesting results.

The **lowess** and **graph twoway lowess** commands both accomplish lowess smoothing (for locally weighted scatterplot smoothing). The **lowess** command with **generate** option can save predicted values. **graph twoway lowess** has advantages of simplicity, and follows the familiar syntax and overlay capabilities of the **graph twoway** family. For a basic example we graph global temperature anomalies using dataset *global3.dta*, introduced in Chapter 2.

```
. use C:\data\global3.dta, clear
. describe

Contains data from c:\data\global3.dta
obs: 1,584                                         Global climate
vars: 5                                              1 Apr 2012 11:35
size: 20,592

variable name  storage   display      value      variable label
              type     format    label
year          int       %8.0g        Year
month         byte      %8.0g        Month
edate         int       %tdmCY      elapsed date
temp          float     %9.0g       NCDC global temp anomaly vs
                                1901-2000, C
mei           float     %9.0g       Multivariate ENSO Index

Sorted by: year month
```

Global temperature anomalies from 1880 through 2011 show considerable variation. From one month or year to the next, it would be impossible to tell whether the longer-term climate is warming, cooling or staying about the same. Lowess smoothing provides a view of longer-term change underneath rapid monthly fluctuations. Figure 8.1 draws a line plot of temperature against elapsed date (**twoway line temp edate**), then overlays this with a lowess smoothed curve

with bandwidth .3, using a thicker line for emphasis (**lowess temp edate, bw(.3) lwidth(thick)**). Type **help linewidthstyle** for other choices of line width.

```
. graph twoway line temp edate
    || lowess temp edate, bw(.3) lwidth(thick)
    || , legend(position(11) ring(0) rows(2))
```

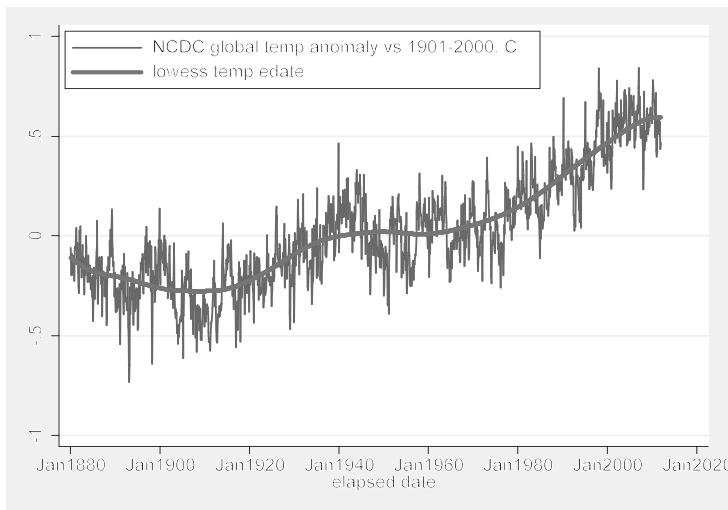


Figure 8.1

The lowess curve in Figure 8.1 clearly shows stages of early 20th century warming (especially 1920–1940), mid-century cooling, and modern rapid warming (post-1970) comprising the signature pattern of global climate change. The same stages appeared in local data on Lake Winnipesaukee ice-out dates, graphed in Figure 3.26 of Chapter 3.

Lowess predicted (smoothed) y values for n observations result from n weighted regressions. Let k represent the half-bandwidth, truncated to an integer. For each y_i , a smoothed value y_i^s is obtained by weighted regression involving only those observations within the interval from $i = \max(1, i - k)$ through $i = \min(i + k, n)$. The j th observation within this interval receives weight w_j according to a tricube function:

$$w_j = (1 - |u_j|^3)^3$$

where

$$u_j = (x_i - x_j) / \Delta$$

Δ stands for the distance between x_i and its furthest neighbor within the interval. Weights equal 1 for $x_i = x_j$, but fall off to zero at the interval's boundaries. See Chambers et al. (1983) or Cleveland (1993) for more discussion and examples of lowess methods.

lowess options include the following.

- mean** For running-mean smoothing. The default is running-line least squares smoothing.
- noweight** Unweighted smoothing. The default is Cleveland's tricube weighting function.

- bwidth()** Specifies the bandwidth. Centered subsets of approximately $bwidth \times n$ observations are used for smoothing, except towards the endpoints where smaller, uncentered bands are used. The default is **bwidth(.8)**.
- logit** Transforms smoothed values to logits.
- adjust** Adjusts the mean of smoothed values to equal the mean of the original y variable; like **logit**, **adjust** is useful with dichotomous y .
- gen(newvar)** Creates *newvar* containing smoothed values of y .
- nograph** Suppresses displaying the graph.
- addplot()** Add other plots to the generated graph; see **help addplot option**.
- lineopts()** Affects the rendition of the smoothed line; see **help cline options**. Because it requires n weighted regressions, lowess smoothing proceeds slowly with large samples.

Like other smoothing methods (or indeed, any model), lowess breaks the data into two parts: a smooth part such as the thick curve in Figure 8.1, and a rough part which is left over after subtracting the smooth from the data. Often the rough contains useful information as well. To illustrate we shift to a different atmospheric dataset on a multi-century timescale, containing measurements from the Greenland Ice Sheet 2 (GISP2) project described in Mayewski, Holdsworth and colleagues (1993) and Mayewski, Meeker and colleagues (1993). Researchers extracted and chemically analyzed an ice core representing more than 100,000 years of climate history. *Greenland_sulfate.dta* includes a very small fraction of this information: measured non-sea salt sulfate concentration and an index of Polar Circulation Intensity since the year 1500.

```
. use C:\data\Greenland_sulfate.dta, clear
. describe

Contains data from C:\data\Greenland_sulfate.dta
    obs:                 271                                     Greenland ice core sulfate &
                                         PCI, 1500-1985 (Mayewski 1993)
    vars:                   3                                     22 Jun 2012 08:47
    size:                4,878

variable name  storage  display      value
           type   format     label      variable label
year          int       %ty        Year
sulfate       double    %10.0g    SO4 ion concentration, ppb
PCI           double    %6.0g     Polar circulation Intensity

Sorted by: year
```

To retain more detail from this 271-point time series, we smooth with a relatively narrow bandwidth, only 5% of the sample. Figure 8.2 graphs the results for *sulfate*.

```
. graph twoway line sulfate year
    || lowess sulfate year, bwidth(.05) lwidth(medthick)
    || , ytitle("SO4 ion concentration, ppb")
    legend(label(1 "raw data") label(2 "lowess smoothed")
    position(11) ring(0) rows(2)))
```

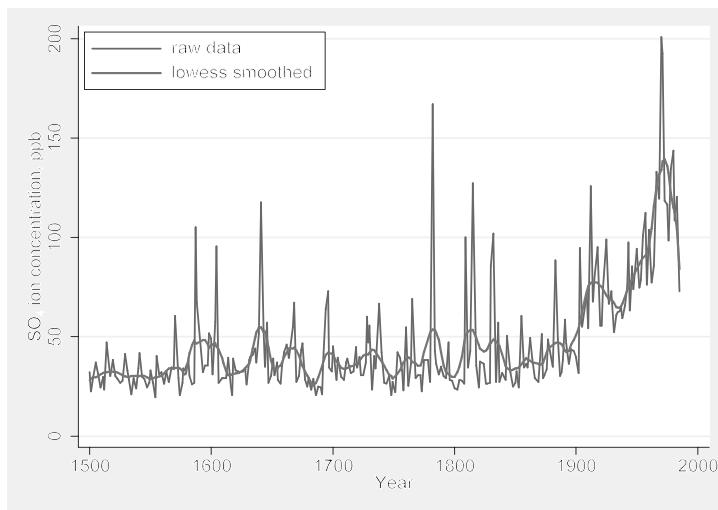


Figure 8.2

Non-sea salt sulfate (SO_4) reached the Greenland ice after being injected into the atmosphere, chiefly by volcanoes or the burning of fossil fuels such as coal and oil. Both the smoothed and raw curves in Figure 8.2 convey information. The smoothed curve shows oscillations around a slightly rising mean from 1500 through the early 1800s. After 1900, fossil fuels drive the smoothed curve upward, with temporary setbacks after 1929 (the Great Depression) and the early 1970s (combined effects of the U.S. Clean Air Act, 1970; the Arab oil embargo, 1973; and subsequent oil price hikes). Most of the sharp peaks of the raw data have been identified with known volcanic eruptions such as Iceland's Hekla (1970) or Alaska's Katmai (1912).

After smoothing time series data, it is often useful to study the smooth and rough (residual) series separately. Below we use the **lowess** command to define two new variables: lowess-smoothed values of sulfate (*smooth*) and the residuals or rough values (*rough*) calculated by subtracting the smoothed values from the raw data.

```
. lowess sulfate year, bwidth(.05) gen(smooth)
. label variable smooth "SO4 ion concentration (smoothed)"
. gen rough = sulfate - smooth
. label variable rough "SO4 ion concentration (rough)"
```

Figure 8.3 compares the *smooth* and *rough* time series in a pair of graphs annotated using **text()** options. Note the use of **saving()** options within each of the first two graph commands. These two graphs are then placed into one image with **graph combine**. In the combined graph, a common *x*-axis title is supplied: **b1title("Year")**. **b1** refers to the first bottom title. A **combined** graph does not know about *x* and *y* axis titles, but recognizes bottom (**b1** and **b2**), left (**l1** and **l2**), top (**t1** and **t2**) or right (**r1** and **r2**) titles instead. In Figure 8.3 a common *y*-axis title is supplied as the first left-hand title, **l1title("SO₄ ion concentration, ppb")**.

```

. graph twoway line smooth year, ylabel(0(50)150) xtitle("")
    lwidth(medthick) lcolor(maroon) ytitle("Smoothed")
    text(20 1540 "Renaissance") text(20 1900 "Industrialization")
    text(90 1860 "Great Depression 1929")
    text(150 1935 "Oil Embargo 1973")
    saving(fig08_03a.gph, replace)

. graph twoway line rough year, ylabel(0(50)150) xtitle("")
    ytitle("Rough") text(75 1630 "Aru 1640",
        orientation(vertical))
    text(120 1770 "Laki 1783", orientation(vertical))
    text(90 1805 "Tambora 1815", orientation(vertical))
    text(65 1902 "Katmai 1912", orientation(vertical))
    text(80 1960 "Hekla 1970", orientation(vertical))
    yline(0) saving(fig08_03b.gph, replace)

. graph combine fig08_03a.gph fig08_03b.gph, rows(2) b1title("Year")
    b2title("SO4 ion concentration, ppb")

```

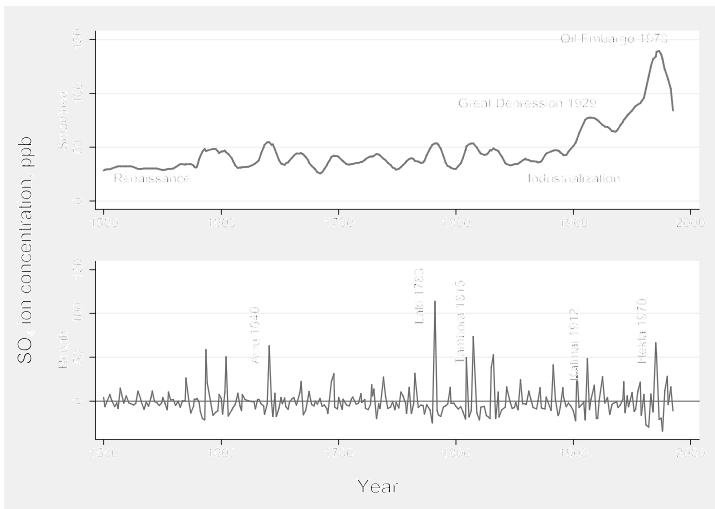


Figure 8.3

Robust Regression

Stata's basic **regress** and **anova** commands perform ordinary least squares (OLS) regression. The popularity of OLS derives in part from its theoretical advantages given ideal data. If errors are normally, independently and identically distributed (normal i.i.d.), then OLS is more efficient than any other unbiased estimator. The flip side of this statement often gets overlooked: if errors are not normal, or not i.i.d., then other unbiased estimators might outperform OLS. In fact, the efficiency of OLS degrades quickly in the face of heavy-tailed (outlier-prone) error distributions. Yet such distributions are common in many fields.

OLS tends to track outliers, fitting them at the expense of the rest of the sample. Over the long run, this leads to greater sample-to-sample variation or inefficiency when samples often contain outliers. Robust regression aims to achieve almost the efficiency of OLS with ideal data and substantially better-than-OLS efficiency in less ideal situations such as nonnormal errors. Robust regression encompasses a variety of different techniques, each with advantages and drawbacks for dealing with problematic data. This section introduces two varieties of robust regression, **rreg** and **qreg**, and briefly compares them with OLS (**regress**).

In Chapter 7 we noted a clear downward trend, steeper than linear, in the minimum area of Arctic sea ice over the years 1979–2011. What about Antarctic sea ice? Its geography and seasonal patterns are quite different. The central Arctic is an ocean surrounded by land, which even after the recent declines retained more than 3 million km² of area, or 4 million km² of extent (the area with at least 15% ice) at its summer minimum. The Antarctic, on the other hand, is land surrounded by ocean. Whereas Arctic sea ice extends to the North Pole, Antarctic sea ice exists at comparatively lower latitudes far from the South Pole. Most of the Antarctic sea ice melts every summer. At the annual minimum in February, Antarctic sea ice area falls to about 2 million km², and extent to less than 3 million km². Dataset *Antarctic2.dta* contains mean February sea ice extent data (Milke and Heygster 2009), covering the years 1972–2011. The dataset also contains annual air temperature anomalies for the Antarctic region, from 64 to 90 degrees south latitude, estimated by NASA.

Contains data from C:\data\Antarctic2.dta				
obs:	39	Antarctic February mean sea ice 1973–2011 Milke & Heygster 2009 29 Apr 2012 16:49		
vars:	4			
size:	429			
variable	storage	display	value	variable label
name	type	format	label	
year	int	%8.0g		Year
yeargrp	byte	%9.0g	dec	1972–1999 v. 2000–2011
extents	float	%9.0g		SH sea ice extent, million km ²
tempS	float	%9.0g		Annual air temp anomaly 64S–90S C

Sorted by: year

Is minimum Antarctic sea ice extent trending up or down? OLS regression finds a weak and statistically nonsignificant downward trend ($p = .125$).

. regress extents year

Source	SS	df	MS	Number of obs = 39
Model	.480675664	1	.480675664	F(1, 37) = 2.46
Residual	7.22814772	37	.195355344	Prob > F = 0.1253
Total	7.70882338	38	.202863773	R-squared = 0.0624
				Adj R-squared = 0.0370
				Root MSE = .44199

extents	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
year	-.0098642	.0062885	-1.57	0.125	-.022606 .0028776
_cons	22.84955	12.52695	1.82	0.076	-2.532471 48.23156

The Arctic decline was obvious in graphs (see Figures 7.9 or 7.12). A graph of the Antarctic decline in Figure 8.4 reveals no obvious trend. We do see potential statistical problems,

however. High *extents* values in 1972 and 1977, a period when satellite observations were less detailed, might be influencing the regression line and causing its weak negative slope.

```
. predict exthat1
. label variable exthat1 "regress (OLS)"
. graph twoway connect extents exthat1 year, msymbol(0 +)
```

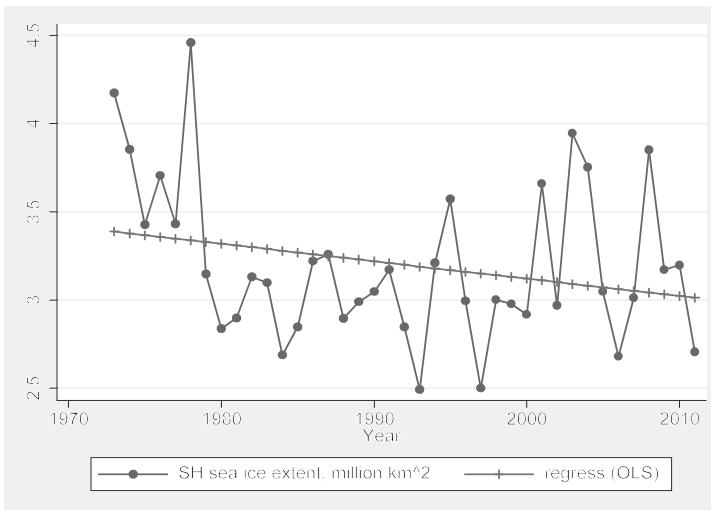


Figure 8.4

Robust regression resists the pull of outliers, making it well suited for a quick check on whether outliers unduly affect our OLS results. The command **rreg** performs robust regression. Applied to Antarctic sea ice, **rreg** finds a decline that is slightly less steep, and also not significant.

```
. rreg extents year
      Huber iteration 1: maximum difference in weights = .61809736
      Huber iteration 2: maximum difference in weights = .10954644
      Huber iteration 3: maximum difference in weights = .0319613
      Biweight iteration 4: maximum difference in weights = .23592582
      Biweight iteration 5: maximum difference in weights = .08565176
      Biweight iteration 6: maximum difference in weights = .02170203
      Biweight iteration 7: maximum difference in weights = .00318406
```

```
Robust regression
Number of obs = 39
F( 1, 37) = 1.62
Prob > F = 0.2105
```

extents	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
year	-.0080677	.0063304	-1.27	0.210	-.0208943 .0047588
_cons	19.21456	12.61029	1.52	0.136	-6.336321 44.76544

After **rreg**, the **predict** command works as usual to obtain predicted values. Graphing these predicted values (here named *exthat2*), Figure 8.5 visually compares robust and OLS lines.

```
. predict exthat2
. label variable exthat2 "rreg (robust)"
. graph twoway connect extents exthat1 exthat2 year,
    msymbol(O + i) lwidth(medium medium thick)
    legend(ring(0) position(12) col(1))
```

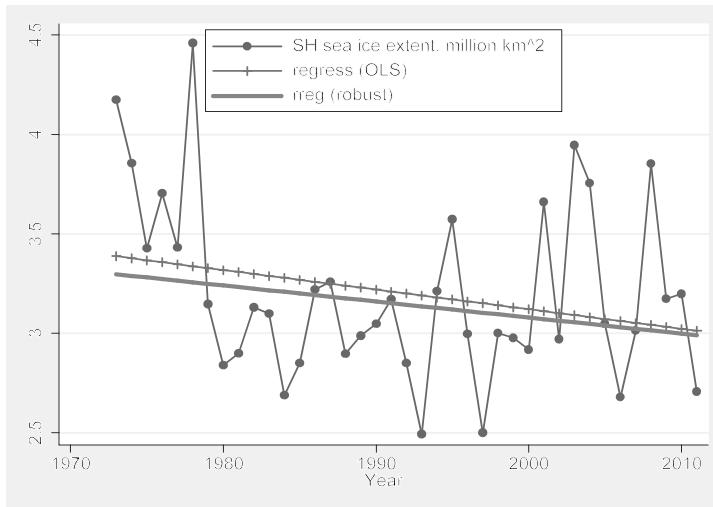


Figure 8.5

rreg works by iteratively reweighted least squares (IRLS). The first **rreg** iteration begins with OLS. Any observations so influential as to have Cook's D values greater than 1 are automatically set aside after this first step. Next, weights are calculated for each observation using a Huber function (which downweights observations that have larger residuals) and weighted least squares is performed. After several WLS iterations, the weight function shifts to a Tukey biweight (as suggested by Li 1985), tuned for 95% Gaussian efficiency (see Hamilton 1992a for details). **rreg** estimates standard errors and tests hypotheses using a pseudovalues method (Street, Carroll and Ruppert 1988) that does not assume normality.

rreg and **regress** both belong to the family of M -estimators (for maximum-likelihood). An alternative order-statistic strategy called L -estimation fits quantiles of y , rather than its expectation or mean. For example, we could model how the median (.5 quantile) of y changes with x . **qreg** (an $L1$ -type estimator) accomplishes such quantile regression. Like **rreg**, **qreg** has good resistance to outliers. However, **qreg** tends to be less efficient, or have higher standard errors, compared to **rreg** with most data. That is the case here: **qreg** finds an even shallower slope, but with greater standard errors. Figure 8.6 graphically compares all three linear models.

```
. qreg extents year

Iteration 1: WLS sum of weighted deviations = 12.830409
Iteration 1: sum of abs. weighted deviations = 14.282429
Iteration 2: sum of abs. weighted deviations = 12.595795
Iteration 3: sum of abs. weighted deviations = 12.59334

Median regression
  Raw sum of deviations 12.99536 (about 3.0977242) Number of obs =      39
  Min sum of deviations 12.59334 Pseudo R2 =      0.0309



| extents | Coef.      | Std. Err. | t     | P> t  | [95% Conf. Interval] |
|---------|------------|-----------|-------|-------|----------------------|
| year    | -0.0056349 | .0076686  | -0.73 | 0.467 | -.0211728 .0099031   |
| _cons   | 14.29918   | 15.276    | 0.94  | 0.355 | -16.65295 45.2513    |



. predict exthat3
. label variable exthat3 "qreg (quantile)"
. graph twoway connect extents exthat1 exthat2 exthat3 year,
    msymbol(O + i i) lwidth(medium medium thick medthick)
    lpattern(solid solid solid dash)
    legend(ring(0) position(12) col(1))
```

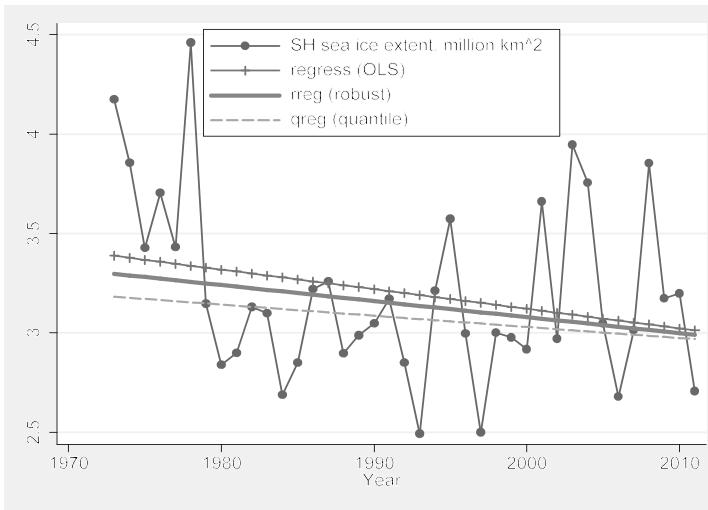


Figure 8.6

qreg performs median regression by default, but has more general capabilities. It can fit linear models for any quantile of y , not just the median (.5 quantile). For example, the following command finds that the third quartile (.75 quantile) of *extents* declines somewhat more steeply than the median does over time. The slope for .75 quantile is -0.0149 , meaning a decline of $14,900 \text{ km}^2$ per year, compared with just $5,600 \text{ km}^2$ per year for the .5 quantile or median. Neither trend is statistically significant, however.

```
. qreg extents year, quant(.75)

Iteration 1: WLS sum of weighted deviations = 12.774128
Iteration 1: sum of abs. weighted deviations = 12.877446
Iteration 2: sum of abs. weighted deviations = 12.702146
Iteration 3: sum of abs. weighted deviations = 12.323326
Iteration 4: sum of abs. weighted deviations = 12.284196

.75 Quantile regression                               Number of obs =      39
  Raw sum of deviations 12.70912 (about 3.4314537)   Pseudo R2      =     0.0334
  Min sum of deviations 12.2842



| extents | Coef.     | Std. Err. | t     | P> t  | [95% Conf. Interval] |
|---------|-----------|-----------|-------|-------|----------------------|
| year    | -.0149048 | .0153885  | -0.97 | 0.339 | -.0460849 .0162753   |
| _cons   | 33.15648  | 30.65445  | 1.08  | 0.286 | -28.95535 95.2683    |


```

Assuming constant error variance, the slopes of the .25 and .75 quantile lines should be roughly the same. **qreg** thus could perform a check for heteroskedasticity or subtle kinds of nonlinearity.

Like **regress**, both **rreg** and **qreg** can work with transformed variables (such as logarithms or squared terms) and with any number of predictors including dummy variables or interactions. Efficiency and ease of use make **rreg** particularly valuable as a quick, general check on whether **regress** results might be affected by outliers or non-normal error distributions. If **rreg** and **regress**, applied to the same model, obtain roughly similar results, then we can state our conclusions with more confidence. Conversely, if **rreg** and **regress** disagree, that presents a yellow flag warning us that our conclusions are unstable. Further investigation is needed to learn why they differ, and figure out what to do about the statistical problem this reveals.

The differences between **regress**, **rreg** and **qreg** results are not large in our Antarctic sea ice example. All three methods agree that there is a weak, statistically nonsignificant downward trend. **regress** gives a slightly steeper slope for this trend because it is more influenced by the high early years. The **rreg** or **qreg** models might be preferred for this reason, but we could also step back and ask whether a linear model is reasonable in the first place. Lowess regression, which does not assume any particular functional form, provides a good tool for exploring questions of this type. Applied to Arctic sea ice (not shown), lowess finds a curve very similar to the quadratic model in the last chapter's Figure 7.12. Applied to Antarctic sea ice in Figure 8.7 below, however, lowess finds nothing resembling a linear or quadratic model. Instead, it suggests a qualitative description of initial decline, subsequent rebound, and perhaps a downturn in the last years. Only the initial decline appears large, and our interpretation there is complicated by limitations of the satellite record. Perhaps a longer-term pattern will become evident in years ahead, but one is not yet clear from these data.

```
. graph twoway connect extents year || lowess extents year
```

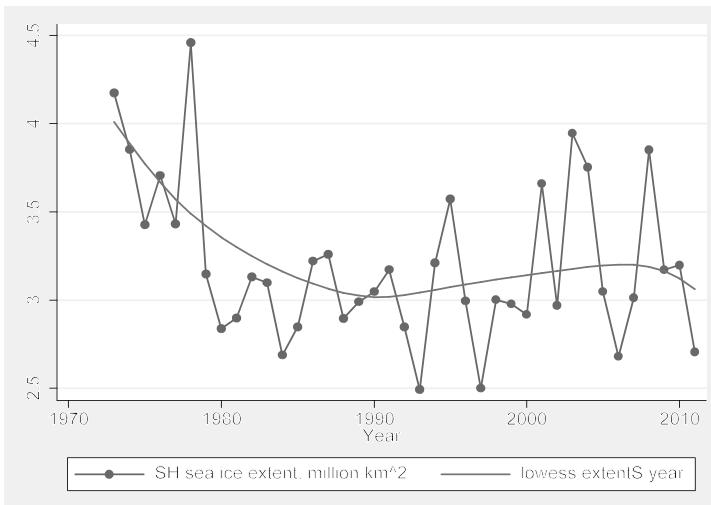


Figure 8.7

Further rreg and qreg Applications

The previous section showed basic applications of **rreg** and **qreg**. These commands also can be used in many other ways, from simple to complex. For example, to obtain a 90% confidence interval for the mean of a single variable such as Antarctic air temperature (*tempS*), we could type the usual confidence-interval command **ci**:

```
. ci tempS, level(90)
```

variable	obs	Mean	Std. Err.	[90% Conf. Interval]
tempS	39	.3351282	.065531	.2246459 .4456105

Alternatively, we could get exactly the same mean and interval through a regression with no *x* variables. The **nohead** option suppresses the top of a regression table, which is not needed here.

```
. regress tempS, nohead level(90)
```

tempS	Coef.	Std. Err.	t	P> t	[90% Conf. Interval]
_cons	.3351282	.065531	5.11	0.000	.2246459 .4456105

Similarly, we can obtain a robust mean with 90% confidence interval. **nolog** suppresses the robust iteration log here to save space.

```
. rreg temps, nolog level(90)
```

Robust regression

Number of obs =	39
F(0, 38) =	0.00
Prob > F =	.

tempS	Coef.	Std. Err.	t	P> t	[90% Conf. Interval]
_cons	.318898	.0707103	4.51	0.000	.1996837 .4381124

qreg could be used in the same way to obtain approximate confidence intervals for one or more medians, with the caveat that a .5 quantile found by **qreg** might not exactly equal a sample median. In theory, .5 quantiles and medians are the same. In practice, quantiles are approximated from actual sample data values, whereas the median is calculated by averaging the two central values, if a subgroup contains an even number of observations. The sample median and .5 quantile approximations then can be different, but in a way that does not much affect model interpretation.

```
. qreg temps, nolog level(90)
```

Median regression

Raw sum of deviations	12.63 (about .28)
Min sum of deviations	12.63

Number of obs =	39
-----------------	----

Pseudo R2 =	0.0000
-------------	--------

tempS	Coef.	Std. Err.	t	P> t	[90% Conf. Interval]
_cons	.28	.0785255	3.57	0.001	.1476096 .4123904

The robust mean is slightly lower than the ordinary mean (.319 versus .335), and the .5 quantile (.28) is lower than either — suggesting that a few warm years are pulling the mean up. In any of these commands, the **level()** option specifies the desired degree of confidence. If we omit this option, Stata automatically displays a 95% confidence interval.

To compare two means, analysts typically employ a two-sample *t* test (**ttest**) or one-way analysis of variance (**oneway** or **anova**). As seen earlier, we can perform equivalent tests (yielding identical *t* and *F* statistics) by regressing the measurement variable on a dummy variable. The dummy variable *yeargrp*, coded 0 for years 1972–1999 and 1 for years 2000–2011, provides an illustration. Figure 8.8 graphs temperature anomalies for these two groups of years.

```
. graph box tempS, over(yeargrp)
```

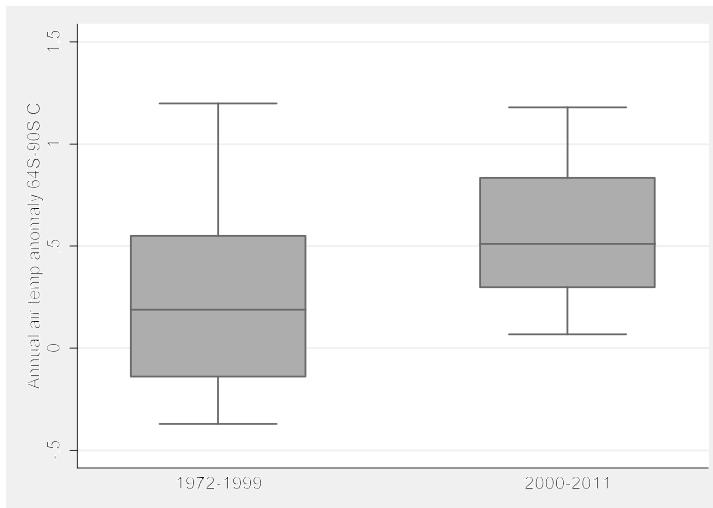


Figure 8.8

Regression analysis confirms the visual impression from Figure 8.8: later years are significantly ($p = .026$) warmer. The mean Antarctic temperature anomaly is $.239^{\circ}\text{C}$ for 1992–1999, and $.312^{\circ}\text{C}$ higher than that ($.239 + .312 = .551^{\circ}\text{C}$) for 2000–2011. It should be noted that even with half a degree of warming, Antarctica remains a very cold place. Temperatures that stay well below freezing year-round at the South Pole.

```
. regress temps yeargrp, nohead
```

temps	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
yeargrp	.3115741	.1344639	2.32	0.026	.0391244 .5840237
_cons	.2392593	.0745871	3.21	0.003	.0881314 .3903871

Is that conclusion robust? **rreg** finds that the robust means differ even more, by $.329^{\circ}\text{C}$. That difference too is significant ($p = .023$).

```
. rreg temps yeargrp, nolog
```

Robust regression

Number of obs = 39
 $F(1, 37) = 5.60$
 Prob > F = 0.0233

temps	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
yeargrp	.3290205	.1390096	2.37	0.023	.0473602 .6106807
_cons	.2155881	.0771087	2.80	0.008	.0593511 .3718251

qreg finds that the .5 quantiles are even farther apart, $.38^{\circ}\text{C}$. This difference is not statistically significant, however ($p = .082$). The larger difference fails to reach statistical significance because the standard errors are much larger in the **qreg** analysis, resulting in a lower t statistic. A larger standard error reflects **qreg**'s relatively low efficiency.

. qreg temps yeargrp, nolog						Number of obs = 39
Median regression						Pseudo R2 = 0.0863
	Raw sum of deviations	12.63 (about .28)				
	Min sum of deviations	11.54				
temps	coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
yeargrp	.38	.2123914	1.79	0.082	-.0503459	.8103459
_cons	.19	.1178136	1.61	0.115	-.048713	.4287129

With effect coding and suitable interaction terms, **regress** can duplicate ANOVA exactly. That is, **regress** followed by the appropriate **test** commands obtains exactly the same R^2 and F test results that would be obtained using **anova**. Predicted values from such regressions equal the group means. **rreg** can perform parallel analyses, testing for differences among robust means instead of ordinary means. Used in similar fashion, **qreg** opens the third possibility of testing for differences among medians. That allows us to fit models analogous to N -way ANOVA or ANCOVA, but involving .5 quantiles or approximate medians instead of the usual means.

Regardless of error distribution shape, OLS remains an unbiased estimator. Over the long run, its estimates should center on the true parameter values. The same is not true for most robust estimators. Unless errors are symmetrical, the median line fit by **qreg**, or the biweight line fit by **rreg**, does not theoretically coincide with the expected- y line estimated by **regress**. So long as the errors' skew reflects only a small fraction of their distribution, **rreg** might exhibit little bias. But when the entire distribution is skewed, **rreg** will downweight mostly one side, resulting in noticeably biased y -intercept estimates. **rreg** slope estimates, however, remain unbiased despite skewed error distributions. Thus there is a tradeoff using **rreg** or similar estimators with skewed errors: we risk getting biased estimates of the y -intercept, but can still expect unbiased and relatively precise estimates of other regression coefficients. In many applications, such coefficients are substantively more interesting than the y -intercept, making the tradeoff worthwhile. Moreover, the robust t and F tests, unlike those of OLS, do not assume normal errors.

Nonlinear Regression — 1

Variable transformations allow fitting some curvilinear relationships using the familiar techniques of intrinsically linear models. Intrinsically nonlinear models, on the other hand, require a different class of fitting techniques. The **nl** command performs nonlinear regression by iterative least squares. This section illustrates with the artificial examples in *nonlin.dta*:

Contains data from c:\data\nonlin.dta				Nonlinear model examples (artificial data)
obs:	100 <th>vars:</th> <td>5</td> <th>29 Apr 2012 18:22</th>	vars:	5	29 Apr 2012 18:22
size:	1,700 <th></th> <th></th> <th></th>			
variable	storage type	display format	value label	variable label
x	byte	%9.0g		Independent variable
y1	float	%9.0g		$y_1 = 10 * 1.03^x + e$
y2	float	%9.0g		$y_2 = 10 * (1 - .95^x) + e$
y3	float	%9.0g		$y_3 = 5 + 25/(1+exp(-.1*(x-50))) + e$
y4	float	%9.0g		$y_4 = 5 + 25*exp(-exp(-.1*(x-50))) + e$

Sorted by: x

The *nonlin.dta* data are manufactured, with *y* variables defined as various nonlinear functions of *x*, plus random Gaussian errors. *y1*, for example, represents the exponential growth process

$$y_1 = 10 \times 1.03^x$$

Estimating these parameters from the data, **nl** obtains

$$y_1 = 11.20 \times 1.03^x$$

which is reasonably close to the true model.

```
. nl (y1 = {b1=1} * {b2=1} ^ x)
(obs = 100)
```

```
Iteration 0: residual SS = 419135.4
Iteration 1: residual SS = 416152.4
Iteration 2: residual SS = 409107.7
Iteration 3: residual SS = 348535.9
Iteration 4: residual SS = 31488.48
Iteration 5: residual SS = 27849.49
Iteration 6: residual SS = 26139.18
Iteration 7: residual SS = 26138.29
Iteration 8: residual SS = 26138.29
Iteration 9: residual SS = 26138.29
```

Source	SS	df	MS	Number of obs = 100
Model	667018.255	2	333509.128	R-squared = 0.9623
Residual	26138.2933	98	266.717278	Adj R-squared = 0.9615
Total	693156.549	100	6931.56549	Root MSE = 16.33148
				Res. dev. = 840.3864

y1	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
/b1	11.20416	1.146683	9.77	0.000	8.928602 13.47971
/b2	1.028838	.0012404	829.41	0.000	1.026376 1.031299

The **predict** command obtains predicted values and residuals for a nonlinear model estimated by **nl**. Figure 8.9 graphs predicted values from the previous example, showing the close fit ($R^2 = .96$) between model and data.

```
. predict yhat1
. graph twoway scatter y1 x
    || line yhat1 x, sort
    || , legend(off) ytitle("y1 = 10 * 1.03^x + e") xtitle("x")
```

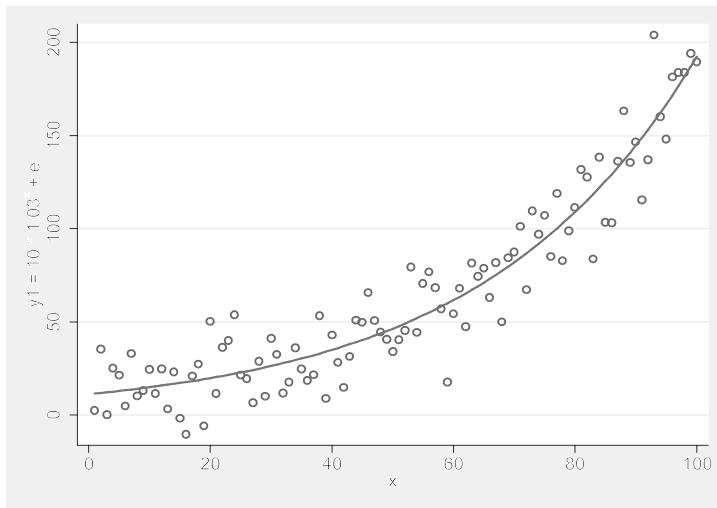


Figure 8.9

Instead of writing out the model in our **nl** command, we could have obtained the same result by typing

```
. nl exp2: y1 x
```

The **exp2** in this command calls a brief program named *nlexp2.ado*, which defines the two-parameter exponential growth function. Stata includes several such programs, for the following functions:

- exp3** 3-parameter exponential: $y = b_0 + b_1 b_2^x$
- exp2** 2-parameter exponential: $y = b_1 b_2^x$
- exp2a** 2-parameter negative exponential: $y = b_1(1 - b_2^x)$
- log4** 4-parameter logistic; b_0 starting level and $(b_0 + b_1)$ asymptotic upper limit:

$$y = b_0 + b_1 / (1 + \exp(-b_2(x - b_3)))$$
- log3** 3-parameter logistic; 0 starting level and b_1 asymptotic upper limit:

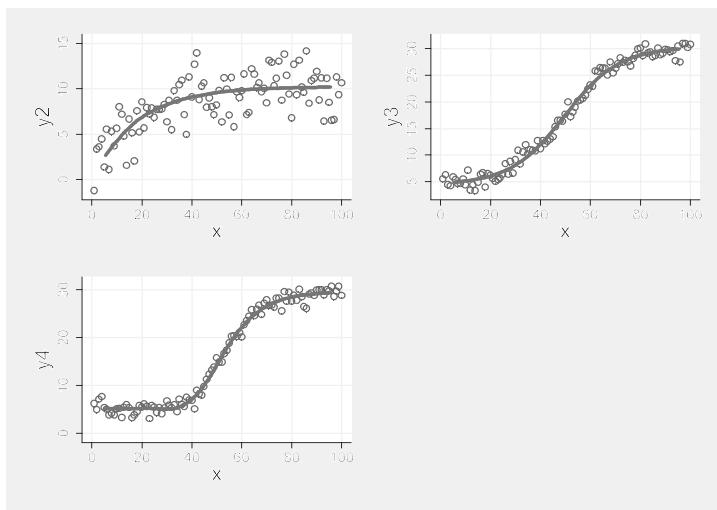
$$y = b_1 / (1 + \exp(-b_2(x - b_3)))$$
- gom4** 4-parameter Gompertz; b_0 starting level and $(b_0 + b_1)$ asymptotic upper limit:

$$y = b_0 + b_1 \exp(-\exp(-b_2(x - b_3)))$$
- gom3** 3-parameter Gompertz; 0 starting level and b_1 asymptotic upper limit:

$$y = b_1 \exp(-\exp(-b_2(x - b_3)))$$

Users can write further **nlfunction** programs of their own; consult *nlexp3.ado*, *nlgom4.ado* or others above for examples. **help nl** describes many further options for specifying and estimating models.

nonlin.dta contains examples corresponding to **exp2** (*y1*), **exp2a** (*y2*), **log4** (*y3*) and **gom4** (*y4*) functions. Figure 8.10 shows curves fit by **nl** to *y2*, *y3* and *y4* from these data.

**Figure 8.10**

Nonlinear Regression — 2

September ice in the Arctic (*Arctic9.dta*) provides a real-world example. Previously we saw that sea ice area declined over 1979–2011, the period of close satellite observation. Diagnostic graphs reveal that a linear model fits poorly, however, because the decline has been faster than linear (Figures 7.9, 7.11). An alternative quadratic model better describes the trend in observations through 2011 (Figure 7.12). If the quadratic model is projected a few years ahead, however, its path becomes physically impossible — crashing rapidly to zero and continuing down to negative area. Physical-process models tend to project a more gradual decline, flattening out as area approaches zero (e.g., Wang and Overland 2009). A simple analogue for such physical models might be an asymmetrical S-curve such as the Gompertz, rather than the precipitous fall of a quadratic.

The following commands fit a 3-parameter Gompertz model for Arctic sea ice. For minor variety we focus on *extent* (at least 15% ice) rather than the *area* (100% ice) used in earlier examples. The two variables have similar behavior, as seen earlier in Figure 3.14.

```
. use C:\data\Arctic9.dta, clear
. nl gom3: extent year, nolog
```

(obs = 33)

Source	SS	df	MS	Number of obs	=	33
Model	1425.43798	3	475.145994	R-squared	=	0.9957
Residual	6.15941312	30	.205313771	Adj R-squared	=	0.9953
Total	1431.5974	33	43.3817393	Root MSE	=	.4531156
3-parameter Gompertz function, extent = b1*exp(-exp(-b2*(year - b3)))						
extent	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
/b1	7.580278	.291652	25.99	0.000	6.984645	8.175911
/b2	-.0995915	.0271646	-3.67	0.001	-.155069	-.044114
/b3	2017.531	2.173212	928.36	0.000	2013.093	2021.969

The Gompertz model fits well, and all three parameters are significant. The first parameter, $b_1 = 7.58$, gives the model's asymptotic starting point, 7.58 million km². The third parameter, $b_3 = 2017.531$, gives the inflection point where this curve shifts from concave upwards (steepening rate of decline) to convex upwards (slowing rate of decline): during the year 2017. The second parameter, $b_2 = -0.0996$, controls change in the rate of decline. To visualize this model, Figure 8.11 graphs the predicted values connected by a median spline (smooth curve).

```
. predict gomext1
. graph twoway connect extent year
    || mspline gomext1 year, band(50) lwidth(medthick)
    legend(label(1 "Observed extent") label(2 "Gompertz prediction"))
```

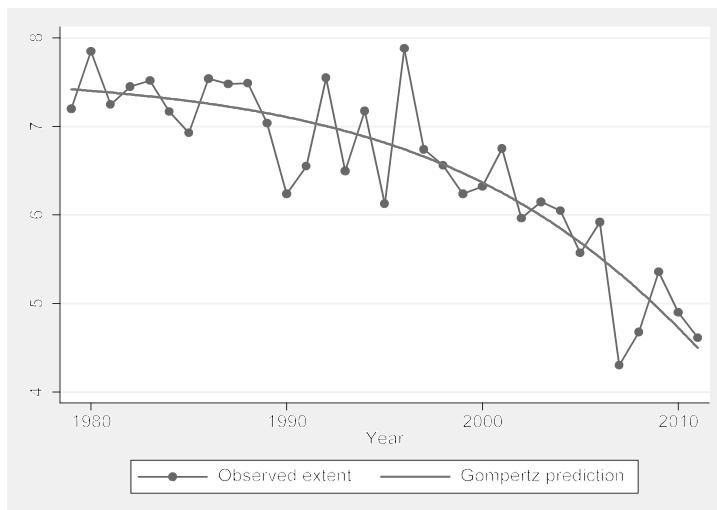


Figure 8.11

The Gompertz curve in Figure 8.11 does not look much different from a quadratic curve (not shown), and fits only slightly better. It does lack the quadratic's unrealistic slight increase in the earliest years. Substantial differences appear, however, if we extrapolate this Gompertz curve beyond the range of the data.

As a purely speculative, if-then exercise, we could extend our analysis to the year 2030. The actual data end with 2011, so we start by adding 19 more observations containing no ice data, but only new *year* values from 2012 to 2030. The first command sets the number of observations at 52 (was 33). The second calculates *year* values equal to the previous year plus one, for each observation that does not yet have a year. Finally we re-estimate the Gompertz model with the new years, obtaining exactly the same result as before.

```
. set obs 52
. replace year = year[_n-1]+1 if year==.
. sort year
. nl gom3: extent year, nolog
(obs = 33)
```

Source	SS	df	MS	Number of obs	=	33
Model	1425.43798	3	475.145994	R-squared	=	0.9957
Residual	6.15941312	30	.205313771	Adj R-squared	=	0.9953
Total	1431.5974	33	43.3817393	Root MSE	=	.4531156
				Res. dev.	=	38.25858

3-parameter Gompertz function, extent = b1*exp(-exp(-b2*(year - b3)))

extent	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
/b1	7.580278	.291652	25.99	0.000	6.984645 8.175911
/b2	-.0995915	.0271646	-3.67	0.001	-.155069 -.044114
/b3	2017.531	2.173212	928.36	0.000	2013.093 2021.969

Although the ice data and the model are unchanged, our extended dataset contains additional years. This makes it possible to obtain predicted values for all years from 1979 to 2030.

```
. predict gomext2
```

Unlike predicted values, residuals are defined only for the years with data on *extent*.

```
. predict res, resid
. summarize res
```

variable	Obs	Mean	Std. Dev.	Min	Max
res	33	.0000746	.4387273	-1.039796	1.137713

The following commands use the predicted extent values, *gomext2*, together with the standard deviation of the residuals, *r(sd)* (results defined by the **summarize** command above) to define lower and upper confidence limits of $\pm 2\text{sd}$ around the predicted values. We then draw a more elaborate graph showing the Gompertz curve extrapolated out to 2030, with emphasis on its prediction for the year 2012.

```
. gen gomlo = gomext2 -2*r(sd)
. gen gomhi = gomext2 +2*r(sd)
. label variable gomext2 "nl gom3: extent year"
. label variable gomlo "Gompertz extent - 2sd"
. label variable gomhi "Gompertz extent + 2sd"
```

```
. graph twoway rarea gomlo gomhi year if year < 2012, color(gs13)
|| mspline gomext2 year if year < 2012, bands(60)
    lwidth(thick) lcolor(maroon)
|| mspline gomext2 year if year >= 2012, bands(60)
    lwidth(medthick) lcolor(maroon) lpattern(dash)
|| connect extent year, lwidth(medthick) msymbol(0)
    lcolor(navy) mcolor(navy)
|| scatter gomext2 year if year == 2012, msymbol(S)
    mcolor(maroon)
|| if year>1978, xlabel(1980(5)2030, grid)
yline(0, lcolor(black)) yline(1, lcolor(gs11) lwidth(thick))
xtitle("") legend(order(4 2 1) label(2 "Gompertz curve")
label(4 "NSIDC extent") label(1 "`=char(177)' 2sd")
position(7) ring(0) col(1))
text(4.5 2019 "2012 Gompertz" "prediction"
"4.3 (3.4`=char(150)'5.1)", color(maroon))
```

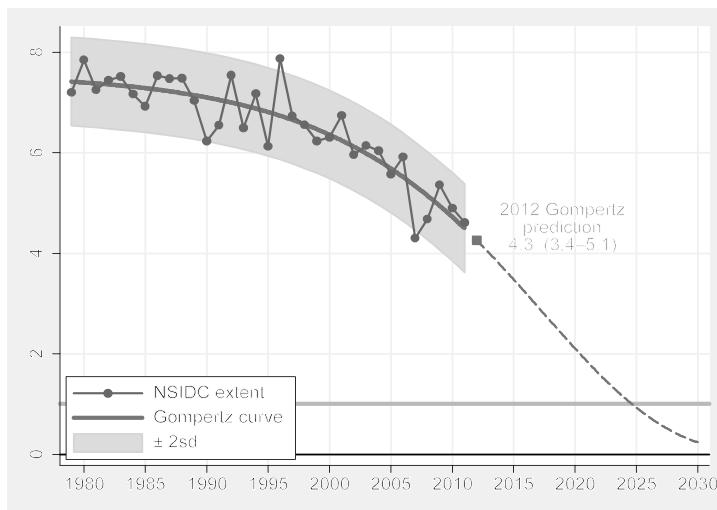


Figure 8.12

The light gray (gs13) confidence bands in Figure 8.12 are drawn first, as a **twoway rarea** (range area) plot for the years before 2012. Next a thick maroon median spline curve (**mspline**) tracing the Gompertz predicted values before 2012 is overlaid on top of the gray confidence bands. A medium-thick dashed curve traces predicted values for years 2012 and later. Observed *extent* values are overlaid next as a connected-line plot (**connect**). The fifth and final overlay consists of a scatterplot with only one point, the predicted value for 2012. Text specifies the numerical value and confidence limits for this prediction. Text color is maroon to match the predicted-values curve it describes. A horizontal gray line at 1 million km² (**yline(1)**) marks a low September level that could be considered virtually ice free.

It must be emphasized that extrapolating curves in this manner is not a reliable way to predict the future. This particular model draws on no physical understanding, such as what might be causing the ice to decline. It depends entirely on our choice of statistical model, and how that

fits past data. As a statistical exercise, however, we can take these naive predictions one more step. The curve in Figure 8.12 drops below 1 million km² line in 2025, offering a point prediction for a virtually ice-free Arctic Ocean. But even if the Gompertz model proved to be true in some sense, we should expect actual behavior to vary around this curve as it has in the past.

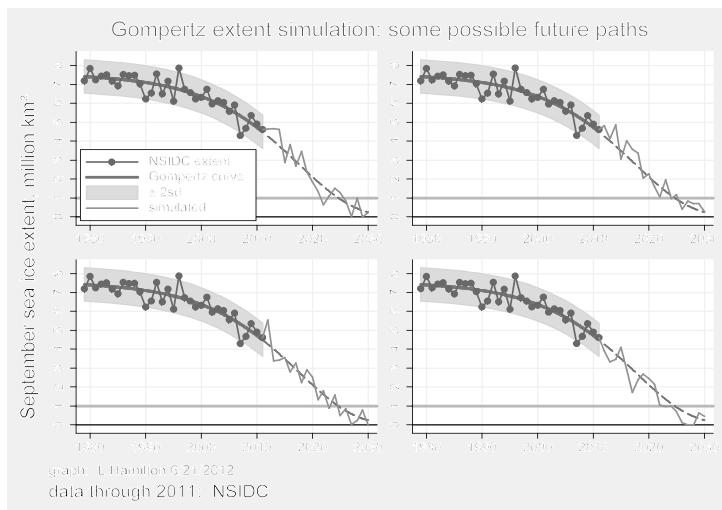
Suppose that variation around the curve in future years followed a normal distribution with the same standard deviation observed around the curve in the past. We could simulate such behavior by adding noise to the smooth curve, drawing random values from a normal distribution with standard deviation equal to that of past residuals. For example, the following commands create a new set of predicted values (*gomext3*) equal to the old predictions (*gomext2*) plus random normal noise with the residuals' standard deviation (*r(sd)* after **summarize res**). If the new prediction suggests a physically impossible negative extent, then it is just set to 0.

```
. quietly summ res
. gen gomext3 = gomext2 + r(sd)*rnormal()
. replace gomext3 = 0 if gomext3 < 0
. replace gomext3 = extent if year == 2011
```

The following commands would graph the results, in a manner similar to Figure 8.11.

```
. graph twoway rarea gomlo gomhi year if year < 2012, color(gs13)
|| mspline gomext2 year if year < 2012, bands(60)
    lwidth(thick) lcolor(maroon)
|| mspline gomext2 year if year >= 2012, bands(60)
    lwidth(medthick) lcolor(maroon) lpattern(dash)
|| connect extent year, lwidth(medthick) msymbol(O)
    lcolor(navy) mcolor(navy)
|| line gomext3 year if year >= 2011, lwidth(medthick)
    lcolor(midblue)
|| , xlabel(1980(10)2030, grid)
yline(0, lcolor(black)) yline(1, lcolor(gs11) lwidth(thick))
ylabel(0(1)8) ytitle("") xttitle("")
legend(order(4 2 1 5) label(2 "Gompertz curve")
    label(4 "NSIDC extent") label(1 "`=char(177)' 2sd")
    label(5 "simulated") position(7) ring(0)
    col(1) rowgap(*.3))
```

Each time we run this set of commands we will get different random noise values and hence a different-looking graph. Figure 8.13 combines four such graphs into one image, which helps to emphasize the unpredictable year-to-year variations. In any given year ice extent might go up or down, while the curve just depicts average behavior that no realization is expected to match.

**Figure 8.13**

Sometimes graphs published on the Internet take on a life of their own and travel far beyond the author's knowledge or intent. For such purposes it can be worthwhile to embed your name, data source or other identifying information within the graphic itself, as done in Figure 8.13 using **note** and **caption** options. The following command draws this composite image by combining four previously-saved graphs, which had been named *Gompertz_extent1* and so forth.

```
. graph combine Gompertz_extent1.gph Gompertz_extent2.gph
    Gompertz_extent3.gph Gompertz_extent4.gph ,
    title("Gompertz extent simulation: some possible future paths",
          size(medlarge))
    caption("data through 2011: NSIDC")
    note("graph: L Hamilton 6/21/2012") imargin(small) col(2)
    l1title("September sea ice extent, million km`=char(178)'")
```

Box–Cox Regression

Leaving colder regions behind, the remainder of this chapter works with the United Nations human development data in *Nations3.dta*. Nonlinear relationships among variables are prominent in scatterplots of these data, such as that in Figure 7.4. Logarithms and other transformations from Tukey's ladder of powers (Chapter 5) provide simple tools for making nonlinear relationships more linear, enabling the application of OLS, robust regression or other linear models.

Selecting which transformation to use might involve trying out several choices, and examining how each affects distributions, scatterplots or residuals. A more systematic approach called Box–Cox regression instead uses maximum likelihood estimation to select Box–Cox transformation parameters that are optimal for a particular regression model. The common Box–Cox transformation is applied to all variables in the model, or any subset of them.

Consider the regression of life expectancy on six other variables including *reg1*, a dummy variable created by **tabulate**, which indicates nations in Africa.

```
. use C:\data\Nations3.dta, clear
. describe life adfert urban gdp chldmort school reg1
```

variable name	storage type	display format	value label	variable label
life	float	%9.0g		Life expectancy at birth 2005/2010
adfert	float	%8.0g		Adolescent fertility: births/1000 fem 15-19, 2010
urban	float	%9.0g		Percent population urban 2005/2010
gdp	float	%9.0g		Gross domestic product per cap 2005\$, 2006/2009
chldmort	float	%9.0g		Prob dying before age 5/1000 live births 2005/2009
school	float	%9.0g		Mean years schooling (adults) 2005/2010
reg1	byte	%8.0g		region==Africa

There is no point in transforming a dichotomous variable like *reg1*, but scatterplots of the other variables show nonlinear patterns that make them candidates for transformation. In this example we regress *life* on *adfert*, *urban*, *gdp*, *chldmort*, *school* and *reg1*. The dependent or left-hand-side variable *life* is left in raw form, but a transformation is sought for all of the right-hand side variables except *reg1*. These choices are specified by options **model(rhsonly)** **notrans(*reg1*)**. Note that initial variable list does not include *reg1*, because it is specified separately in the **notrans()** option.

```
. boxcox life adfert urban gdp chldmort school, model(rhsonly)
    notrans(reg1)
```

Fitting full model

```
Iteration 0: log likelihood = -455.39883 (not concave)
Iteration 1: log likelihood = -430.26519
Iteration 2: log likelihood = -429.92904
Iteration 3: log likelihood = -429.92798
Iteration 4: log likelihood = -429.92798
(15 missing values generated)
(1 missing value generated)
(6 missing values generated)
(15 missing values generated)
(1 missing value generated)
(6 missing values generated)
(15 missing values generated)
(1 missing value generated)
(6 missing values generated)
```

```
Number of obs      =      178
LR chi2(7)        =      463.38
Prob > chi2       =      0.000
Log likelihood = -429.92798
```

life	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
/lambda	.4867359	.0513717	9.47	0.000	.3860492 .5874226

Estimates of scale-variant parameters

	Coef.
Notrans	
reg1	-2.863907
_cons	86.17721
Trans	
adfert	-.0383667
urban	.2065436
gdp	.000283
chldmort	-1.42784
school	-1.601755
/sigma	2.708479

Test	Restricted log Likelihood	LR statistic	P-value
H0:		chi2	Prob > chi2
lambda = -1	-524.20312	188.55	0.000
lambda = 0	-476.81642	93.78	0.000
lambda = 1	-455.39883	50.94	0.000

The lambda value given in this output, $\lambda = .4867359$, is the parameter chosen for Box–Cox transformations of the general form

$$x^{(\lambda)} = \{x^\lambda - 1\} / \lambda$$

The regression coefficients shown in the Box–Cox output above correspond to an ordinary regression with variables transformed in this fashion. We could replicate the Box–Cox regression results by generating transformed versions of each variable, then using **regress**.

```
. gen bc2adf = (adfert^.4867359-1)/.4867359
. gen bc2urb = (urban^.4867359-1)/.4867359
. gen bc2school = (school^.4867359-1)/.4867359
. gen bc2gdp = (gdp^.4867359-1)/.4867359
. gen bc2chld = (chldmort^.4867359-1)/.4867359
. regress life bc2* reg1
```

Source	SS	df	MS			
Model	16332.407	6	2722.06783	Number of obs =	178	
Residual	1305.78282	171	7.63615682	F(6, 171) =	356.47	
Total	17638.1898	177	99.6507898	Prob > F =	0.0000	
				R-squared =	0.9260	
				Adj R-squared =	0.9234	
				Root MSE =	2.7634	
life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
bc2adf	-.0383667	.0598489	-0.64	0.522	-.1565045	.0797711
bc2urb	.2065437	.0955322	2.16	0.032	.0179693	.395118
bc2school	-1.601755	.3209716	-4.99	0.000	-2.235331	-.9681777
bc2gdp	.000283	.0035913	0.08	0.937	-.006806	.0073721
bc2chld	-1.42784	.0792446	-18.02	0.000	-1.584263	-1.271416
reg1	-2.863907	.674814	-4.24	0.000	-4.195945	-1.531869
_cons	86.17721	1.909628	45.13	0.000	82.40773	89.94669

Box–Cox regression finds a transformation parameter λ that is optimal in terms of a maximum-likelihood criterion. Meeting this criterion does not necessarily mean that relationships have been linearized, however. The latter goal might still be more effectively pursued using judgment and visual inspection, with the possibility of different transformations for different variables.

Multiple Imputation of Missing Values

Nations3.dta contains information about 194 countries, but missing values restrict our analysis in previous sections to a subset of 178 that have complete information on all variables of interest. This *listwise deletion* approach of setting aside incomplete observations is, out of necessity, a common statistical practice. Its known drawbacks include loss of observations and statistical power. If the observations with missing values differ systematically from other observations, listwise deletion could also bias coefficient estimates.

There may be other variables in the data that are statistically related to those with missing values. In such cases regression could be used to predict what the missing values might be, and those predictions substituted for the missing values in further analytical steps. This *regression imputation* of missing values can restore observations and apparent statistical power, and reduce the likelihood of biased coefficients. However, the imputed values will generally have lower variance than non-missing values for that variable, leading to standard error estimates that are biased toward zero. In other words, regression imputation may cause us to over-estimate the precision or statistical significance of our results.

Multiple imputation of missing values starts from the core idea of regression imputation, then adds further steps to obtain more realistic estimates of standard errors or uncertainty. These involve creation of multiple sets of artificial observations in which missing values are replaced by regression predictions plus random noise. Then a final step pools the information of these multiple imputations to estimate the regression model along with its standard errors and tests.

Stata's **mi** family of multiple-imputation procedures supports a variety of data organizations, estimation methods and modeling techniques including logit-type models for categorical variables. The *Stata Multiple-Imputation Reference Manual* covers the choices in 365 pages, and could well be supplemented by a companion volume filled with more examples.

For a basic example, we revisit the life expectancy regression.

```
. use C:\data\Nations3.dta, clear
. regress life adfert urban loggdp chldmort school reg1
```

Source	SS	df	MS	Number of obs	=	178
Model	15810.6966	6	2635.1161	F(6, 171)	=	246.57
Residual	1827.49317	171	10.6870946	Prob > F	=	0.0000
Total	17638.1898	177	99.6507898	R-squared	=	0.8964
				Adj R-squared	=	0.8928
				Root MSE	=	3.2691
life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
adfert	-.0039441	.0091498	-0.43	0.667	-.0220053	.014117
urban	.0397506	.016287	2.44	0.016	.0076012	.0718999
loggdp	2.90728	.9196223	3.16	0.002	1.092007	4.722554
chldmort	-.1314439	.0102063	-12.88	0.000	-.1515905	-.1112972
school	-.3322321	.1480558	-2.24	0.026	-.6244844	-.0399798
reg1	-3.56938	.7845902	-4.55	0.000	-5.118109	-2.02065
_cons	65.3779	3.124978	20.92	0.000	59.2094	71.5464

Three of the variables in this analysis — *loggdp*, *chldmort* and *school* — have missing values that in combination reduce the available sample from 194 to 178 observations.

```
. summarize life adfert urban loggdp chldmort school reg1
```

variable	Obs	Mean	Std. Dev.	Min	Max
life	194	68.7293	10.0554	45.85	82.76666
adfert	194	51.81443	44.06612	1	207.1
urban	194	55.43488	23.4391	10.25	100
loggdp	179	3.775729	.5632902	2.446848	4.874516
chldmort	193	47.65026	52.8094	2.25	209
school	188	7.45922	2.959589	1.15	12.7
reg1	194	.2680412	.4440852	0	1

The **misstable summarize** command counts three types of observations, depending on their missing-value status:

- obs = . Stata's default missing value, referred to as "soft missing."
- obs>. Missing value codes shown as .a, .b, .c etc. which could take value labels, referred to as "hard missing."
- obs<. Nonmissing values.

Stata can impute only soft missing values, and not the hard missing ones. All missing values in the *Nations3.dta* example are soft, so the situation is simple. A survey example with hard missing values will be considered in Chapter 9.

```
. misstable summarize life adfert urban loggdp chldmort school reg1
          Obs<.
```

variable	Obs=.	Obs>.	Obs<.	unique values		
				Min	Max	
loggdp	15		179	179	2.446848	4.874516
chldmort	1		193	144	2.25	209
school	6		188	165	1.15	12.7

The first step in multiple imputation is to declare the data using an **mi set** command, which specifies how imputed values will be organized. There are four possible styles, described in the *Reference Manual*. For this example we choose the memory-efficient style **mlong**, in which new observations or rows will be added to the data.

```
. mi set mlong
```

In multiple-imputation notation, the original un-imputed data with missing values are denoted $m = 0$. Imputations with sets of filled-in missing values are denoted $m = 1, m = 2, m = 3$ and so forth. M denotes how many such imputations are done. Before proceeding further we need to register the variables of interest as one of three types.

- imputed** Has missing values to be imputed.
- passive** A variable that is a function of imputed variables or of other passive variables. It will have missing values in the original data ($m = 0$), and varying values in each imputation ($m = 1, m = 2$ etc.).
- regular** Neither imputed nor passive, a variable that has the same values (missing or not) across all m .

Our example has missing values for *loggdp*, *chldmort* and *school*, so these variables we register as **imputed**.

```
. mi register imputed loggdp chldmort school
```

The other variables *life*, *adfert*, *urban* and *reg1* contain no missing values and should be registered as **regular**.

```
. mi register regular life adfert urban reg1
```

The next step does actual imputation. Missing values of *loggdp*, *chldmort* and *school* (the **mi register imputed** variables) are imputed by regression on the **mi register regular** variables *reg1*, *adfert* and *urban*. We use a multivariate normal (**mvn**) regression method. There will be 50 separate imputations denoted as $m = 0$ (the original data with missing values) or $m = 1$ through $m = 50$, each of which contains imputations for the 16 observations that originally had missing values. Thus, $50 \times 16 = 800$ observations will be added to the data, making a total of $194 + 800 = 964$ observations.

```
. mi impute mvn loggdp chldmort school = adfert urban reg1,
    add(50) rseed(12345)
```

Performing EM optimization:
observed log likelihood = -780.80745 at iteration 6

Performing MCMC data augmentation ...

Multivariate imputation	Imputations =	50
Multivariate normal regression	added =	50
Imputed: m=1 through m=50	updated =	0
Prior: uniform	Iterations =	5000
	burn-in =	100
	between =	100

variable	Observations per m			
	Complete	Incomplete	Imputed	Total
loggdp	179	15	15	194
chldmort	193	1	1	194
school	188	6	6	194

(complete + incomplete = total; imputed is the minimum across m of the number of filled-in observations.)

Having 50 separate imputations, each with the missing values replaced, provides a basis for later estimating the sample-to-sample variation when we pool these values for regression. The **rseed(12345)** option in this **mi impute** command sets an arbitrary seed for Stata's random-number generator. By using **rseed()** we make the example repeatable, which might be desirable for instructional purposes. Otherwise Stata will chose its own seed, giving slightly different results the next time the same command is given.

A final step uses these imputations to regress life expectancy on the 6 predictors. In principle the imputation process results in estimates that are more efficient (lower standard errors) and less biased than our earlier regression which dropped all observations with missing values.

```
. mi estimate, dots: regress life adfert urban loggdp chldmort
    school reg1

Imputations (50):
.....10.....20.....30.....40.....50 done

Multiple-imputation estimates
Linear regression
Imputations = 50
Number of obs = 194
Average RVI = 0.0365
Largest FMI = 0.1167
Complete DF = 187
DF: min = 156.85
      avg = 172.10
      max = 184.02

DF adjustment: Small sample
F( 6, 184.7) = 240.19
Prob > F = 0.0000

Model F test: Equal FMI
Within VCE type: OLS


```

life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
adfert	-.0044855	.0091363	-0.49	0.624	-.0225124 .0135414
urban	.047157	.0163199	2.89	0.004	.014945 .079369
loggdp	2.704804	.9755022	2.77	0.006	.7779876 4.63162
chldmort	-.1305611	.0102536	-12.73	0.000	-.1507939 -.1103283
school	-.3317234	.1527427	-2.17	0.031	-.6332567 -.0301902
reg1	-3.814368	.8011135	-4.76	0.000	-5.394917 -2.23382
_cons	65.78153	3.297027	19.95	0.000	59.27041 72.29265

These **mi estimate** results closely resemble those from our earlier ordinary regression. That presents a best-case scenario in which simple and more complicated methods agree: the findings appear reasonably stable. In a research report, we could present either analysis along with a footnote explaining that we had tested another approach as well, and reached the same conclusion.

Chapter 9 presents a second example of multiple imputation, using survey data and logistic instead of linear regression models. Consult **help mi** and the *Stata Multiple-Imputation Reference Manual* for more on this topic.

Structural Equation Modeling

The regression models above treat adolescent fertility, percent urban and other national characteristics as possible predictors of life expectancy, without necessarily asserting that they are its causes. One of those predictors, child mortality rate, has an obvious causal connection with life expectancy. But child mortality rates in turn are influenced by other national characteristics much as life expectancy is, complicating the causal picture. For example, if adolescent fertility rates affect child mortality and child mortality affects life expectancy, then adolescent fertility exerts an indirect effect on life expectancy whether or not it has a distinct effect directly. In causal terms, child mortality functions as an intervening or mediating variable.

Structural equation modeling provides a systematic way to analyze such indirect effects, along with other kinds of causal relationships. Its signature device is the path diagram, which visualizes our ideas about causal ordering and connections. The causal ordering of variables is critical. Structural equation modeling cannot prove causality, but rather assumes a particular causal structure. It then applies statistical techniques to fill in details, fine-tuning the

specification in some respects. We must rely on external knowledge or theory to specify the basic causal order, however. If that knowledge is shaky, then so are the analyses that follow. But sketching path diagrams is a useful step even when the ordering remains uncertain. Often the diagrams help to clarify fuzzy thinking, or better express our ideas.

Approaches based on structural equation modeling have become dominant in diverse areas of research. Since their early applications in the 1960s, structural equation models appealed to social scientists in particular because they promised to bridge worlds of theory and data. The models have been the focus of many books covering issues such as estimation, measurement models, error structures and reciprocal causation (e.g., Kline 2010; Skrondal and Rabe-Hesketh 2004). With version 12, Stata added its own structural equation modeling commands. To quote the *Structural Equation Modeling Reference Manual*:

“SEM is not just an estimation method for a particular model in the way that Stata’s **regress** and **probit** commands are, or even in the way that **stcox** and **xtmixed** are. SEM is a way of thinking, a way of writing, and a way of estimating.”

This section introduces Stata’s structural equation modeling command, **sem**, through a simple extension of our life expectancy regression. We have seen that life expectancy is predicted by other national characteristics. In the regression table below, the *t* statistics and beta weights or standardized regression coefficients (right-hand column) reveal that child mortality rate has by far the strongest effect.

```
. use C:\data\Nations3.dta, clear
. regress life adfert urban loggdp chdmort school reg1, beta
```

Source	SS	df	MS	Number of obs =	178
Model	15810.6966	6	2635.1161	F(6, 171) =	246.57
Residual	1827.49317	171	10.6870946	Prob > F =	0.0000
Total	17638.1898	177	99.6507898	R-squared =	0.8964
				Adj R-squared =	0.8928
				Root MSE =	3.2691

life	Coef.	Std. Err.	t	P> t	Beta
adfert	-.0039441	.0091498	-0.43	0.667	-.0176989
urban	.0397506	.016287	2.44	0.016	.0906915
loggdp	2.90728	.9196223	3.16	0.002	.163546
chdmort	-.1314439	.0102063	-12.88	0.000	-.7013779
school	-.3322321	.1480558	-2.24	0.026	-.0990496
reg1	-3.56938	.7845902	-4.55	0.000	-.1611557
_cons	65.3779	3.124978	20.92	0.000	.

In **sem** notation we could fit the same model as follows:

```
. sem (life <- adfert urban loggdp chldmort school reg1), standard
(16 observations with missing values excluded;
 specify option 'method(mlmv)' to use all observations)

Endogenous variables
Observed: life

Exogenous variables
Observed: adfert urban loggdp chldmort school reg1

Fitting target model:

Iteration 0:  log likelihood = -3455.5273
Iteration 1:  log likelihood = -3455.5273

Structural equation model                         Number of obs      =      178
Estimation method    = ml                      Log Likelihood   = -3455.5273


```

Standardized	OIM				
	coef.	Std. Err.	z	P> z	[95% Conf. Interval]
Structural					
life <-					
adfert	-.0176989	.0402413	-0.44	0.660	-.0965704 .0611725
urban	.0906915	.0363517	2.49	0.013	.0194434 .1619395
loggdp	.163546	.0505432	3.24	0.001	.0644831 .262609
chldmort	-.7013779	.050479	-13.89	0.000	-.8003149 -.6024408
school	-.0990496	.0431943	-2.29	0.022	-.1837089 -.0143903
reg1	-.1611557	.0344906	-4.67	0.000	-.228756 -.0935554
_cons	6.56771	.3476428	18.89	0.000	5.886343 7.249077
Variance					
e.life	.10361	.0109235		.0842675	.1273923

LR test of model vs. saturated: chi2(0) = 0.00, Prob > chi2 = .

The standardized coefficients in this **sem** output equal the beta weights obtained by **regress**. Unlike **regress**, **sem**, reports standard errors for the standardized coefficients. These result in *z* statistics identical to the *t* statistics shown by **regress** but with slightly different probabilities because they are compared to standard normal rather than *t* distributions. The **sem** command's parenthetical notation (*life <- adfert urban loggdp chldmort school reg1*) specifies causal paths to *life* from *adfert*, *urban* etc.

Child mortality, the strongest predictor of life expectancy, might be predicted from many of the same national characteristics. Adolescent fertility, which shows no significant effect on life expectancy in the **regress** or single-equation **sem** output above, turns out to be the strongest single predictor of child mortality.

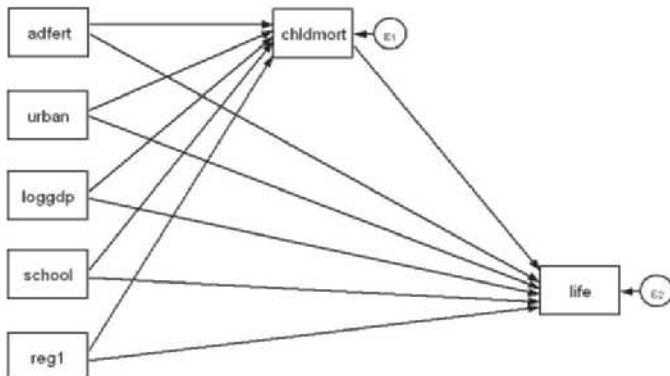
```
. regress chldmort adfert urban loggdp school reg1, beta
```

Source	SS	df	MS	Number of obs	=	178
Model	399607.346	5	79921.4693	F(5, 172)	=	133.99
Residual	102593.515	172	596.473922	Prob > F	=	0.0000
Total	502200.861	177	2837.293	R-squared	=	0.7957
				Adj R-squared	=	0.7898
				Root MSE	=	24.423

chldmort	Coef.	Std. Err.	t	P> t	Beta
adfert	.4319515	.0598983	7.21	0.000	.3632622
urban	-.1123517	.1213743	-0.93	0.356	-.0480386
loggdp	-19.27752	6.711211	-2.87	0.005	-.2032322
school	-3.180865	1.079173	-2.95	0.004	-.1777234
reg1	31.53801	5.345499	5.90	0.000	.2668552
_cons	118.4609	21.52788	5.50	0.000	.

Figure 8.14 shows a path diagram in which child mortality appears as an intervening variable affected by adolescent fertility and other background characteristics, but is itself also a predictor of life expectancy. Conceptually, causality flows from left to right in this diagram. Indirect effects could follow any of the paths from background variables to *chldmort*, and then from *chldmort* to *life*. The boxes represent observed variables in this model; the topic of unobserved or latent variables is raised later, in Chapter 12.

Figure 8.14



The diagram in Figure 8.14 is drawn by a graphical user interface (GUI) called the SEM Builder. This GUI is invoked by typing the command

```
. sembuilder
```

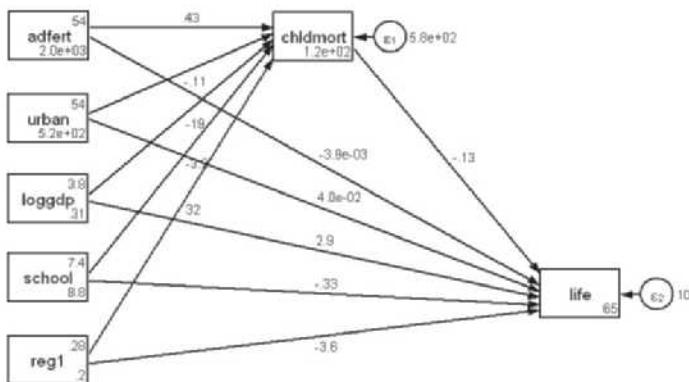
or by menu choices

Statistics > SEM (structural equation modeling) > Model building and estimation

Type **help sembuilder** for information on how to get started. Basic elements in Figure 8.14 are the observed variables, first set in the diagram as empty boxes using the Add Observed Variable tool from the SEM Builder's left-margin menu bar. Next the variable names are filled in by selecting each box with the Select tool, and filling in names from a pull-down Variable menu. An Add Path tool can draw in the paths connecting variables: click on the cause or exogenous variable, then drag the path arrow to connect with an effect or endogenous variable.

From the top menu bar, selecting Estimation > Estimate obtains coefficients and other statistics for the model. Figure 8.15 shows the default result of estimating Figure 8.14. Each path is associated with an unstandardized path or regression coefficient, such as .43 for the effect of *adfert* on *chldmort* (compare with the regression table above).

Figure 8.15



Within the box of each exogenous variable in Figure 8.15 we have the variable's mean and its variance. Among the 178 nations used in this analysis, the mean *adfert* is approximately 54, and the variance is approximately $2.0e+03 = 2,000$. The endogenous variable boxes contain their *y*-intercepts, such as $1.2e+02 = 120$ for *chldmort*; again, compare with the previous regression results. Finally, Figure 8.15 gives residual variances associated with the error terms ϵ_1 (*chldmort*) and ϵ_2 (*life*).

A simplified version of this path diagram containing only standardized path coefficients and standardized residual variances appears in Figure 8.16. The simplification is obtained by making choices from the SEM Builder top menus:

- Settings > Variables > All ... > Results > Exogenous variables > None > OK
- Settings > Variables > All ... > Results > Endogenous variables > None > OK
- Settings > Variables > Error ... > Results > Error std. variance > OK
- Settings > Connections > Paths > Results > Std. parameter > OK
- Settings > Connections > All > Results > Result 1 > Format %3.2f > OK > OK
- Estimation > Estimate > OK

Figure 8.16

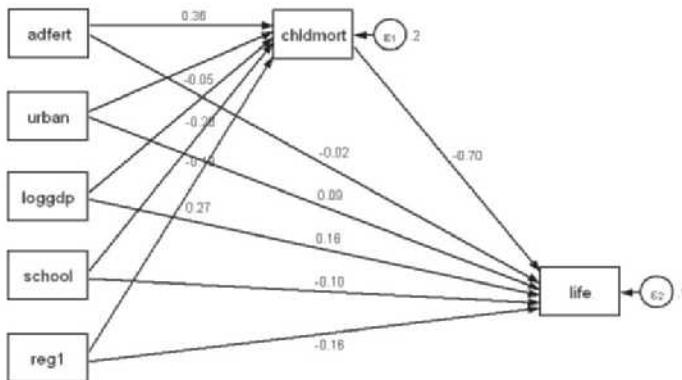


Figure 8.16 corresponds to the following **sem** command, which has separate equations for *life* and *chldmort*.

```

. sem (life <- adfert urban loggdp chldmort school reg1)
      (chldmort <- adfert urban loggdp school reg1), standardized

(16 observations with missing values excluded;
 specify option 'method(mlmv)' to use all observations)

Endogenous variables
Observed: life chldmort
Exogenous variables
Observed: adfert urban loggdp school reg1
Fitting target model:
Iteration 0:  log likelihood = -3455.5273
Iteration 1:  log likelihood = -3455.5273
  
```

```

Structural equation model                               Number of obs      =      178
Estimation method = ml                             Pseudo R-squared   =     0.000
Log Likelihood   = -3455.5273                      Prob > 0.5        =     1.000
  
```

Standardized	Coef.	OIM		z	P> z	[95% Conf. Interval]
		Std. Err.				
Structural						
life <-						
chldmort	-.7013779	.0510955	-13.73	0.000	-.8015233	-.6012325
adfert	-.0176989	.0402432	-0.44	0.660	-.0965741	.0611762
urban	.0906915	.0364057	2.49	0.013	.0193376	.1620454
loggdp	.163546	.0506695	3.23	0.001	.0642356	.2628564
school	-.0990496	.0432485	-2.29	0.022	-.1838151	-.014284
reg1	-.1611557	.03467	-4.65	0.000	-.2291076	-.0932038
_cons	6.56771	.376114	17.46	0.000	5.83054	7.30488

chldm~t <-						
adfert	.3632622	.0478018	7.60	0.000	.2695725	.456952
urban	-.0480386	.0509857	-0.94	0.346	-.1479688	.0518915
loggdp	-.2032322	.0691733	-2.94	0.003	-.3388094	-.0676551
school	-.1777234	.0589333	-3.02	0.003	-.2932306	-.0622162
reg1	.2668552	.0434367	6.14	0.000	.1817208	.3519895
_cons	2.230213	.3963617	5.63	0.000	1.453359	3.007068
<hr/>						
Variance						
e.life	.10361	.0118253			.0828423	.129584
e.chldmort	.2042878	.0211978			.1666933	.2503611

LR test of model vs. saturated: chi2(0) = 0.00, Prob > chi2 = .

Indirect and total effects are easily calculated by hand. Indirect effects equal the product of coefficients along any series of causal paths that link one variable to another. Total effects equal the sum of all direct and indirect effects linking two variables. Reading standardized coefficients from Figure 8.16, adolescent fertility affects life expectancy as follows:

$$\begin{aligned}\text{direct:} \quad & -.02 \\ \text{indirect:} \quad & .36 \times (-.70) = -.25 \\ \text{total:} \quad & -.02 - .25 = -.27\end{aligned}$$

In other words, this model predicts that, other things being equal, a one standard deviation increase in adolescent fertility lowers the predicted life expectancy by .27 standard deviations through direct and indirect effects. The direct effect of *adfert* is near-zero, but it is causally important due to indirect effect through child mortality.

By similar calculations the influence of African location (standing in for troubles of that region not captured by other variables in the model) is more than double the direct effect of *reg1*:

$$\begin{aligned}\text{direct:} \quad & -.16 \\ \text{indirect:} \quad & .27 \times (-.70) = -.19 \\ \text{total:} \quad & -.16 - .19 = -.35\end{aligned}$$

We return for a second look at structural equation modeling in the context of factor analysis, in Chapter 12.

Logistic Regression

The regression methods described in Chapters 7 and 8 generally require measured dependent variables. Stata also offers a full range of techniques for modeling categorical, ordinal and censored dependent variables. The following list gives some idea of the variety of methods available. For more details on specific commands, type **help command**. Long and Freese (2006) provide an excellent source on Stata's main methods for limited dependent variables; also see Hosmer and Lemeshow (2000).

- asclogit** Alternative-specific conditional logit (McFadden's choice)
- asmprobit** Alternative-specific multinomial probit regression.
- asroprobit** Alternative-specific rank-ordered probit regression.
- biprobit** Bivariate probit regression.
- binreg** Binomial regression (generalized linear models).
- blogit** Logit estimation with grouped (blocked) data.
- bprobit** Probit estimation with grouped (blocked) data.
- clogit** Conditional fixed-effects logistic regression.
- cloglog** Complementary log-log estimation.
- constraint** Defines, lists, and drops linear constraints.
- exlogistic** Exact logistic regression.
- glm** Generalized linear models. Includes option to model logistic, probit, or complementary log-log links. Allows response variable to be binary or proportional for grouped data.
- glogit** Logit regression for grouped data.
- gprobit** Probit regression for grouped data.
- heckprob** Probit estimation with selection.
- hetprob** Heteroskedastic probit estimation.
- intreg** Interval regression, where y is either point data, interval data, left-censored data, or right-censored data.
- ivprobit** Probit with continuous exogenous regressors.
- logistic** Logistic regression, giving odds ratios.
- logit** Logistic regression — similar to **logistic**, but giving coefficients instead of odds ratios.
- mlogit** Multinomial logistic regression, for polytomous y variables.
- nlogit** Nested logit estimation.
- ologit** Logistic regression for ordinal y variables.
- oprobit** Probit regression for ordinal y variables.
- probit** Probit regression, for dichotomous y variable.
- rologit** Rank-ordered logit model for rankings (also known as the Plackett–Luce model, exploded logit model, or choice-based conjoint analysis).

- scobit** Skewed probit estimation.
- slogit** Stereotype logistic regression.
- svy: logit** Logistic regression with complex survey data. Survey (**svy**) versions of many other categorical-variable modeling commands also exist (**help svy estimation**).
- tobit** Tobit regression, assuming y follows a Gaussian distribution but is censored at a known, fixed point (see **help cnreg** for a more general version).
- xtmelogit** Binary logit mixed or multilevel models, with fixed and random effects. Type **help xtmelogit** for more about this powerful new command, which is illustrated in Chapter 13. Many other panel-data commands exist; type **help xt** for a list.

After most model-fitting commands, **predict** can calculate predicted values or probabilities. **predict** also obtains diagnostic statistics, such as those described for logistic regression in Hosmer and Lemeshow (2000). Specific **predict** options depend on the type of model just fitted. A different post-fitting command, **predictnl**, obtains nonlinear predictions and their confidence intervals (see **help predictnl**). The **margins** and **marginsplot** commands often prove useful as well.

Examples of several of these commands appear in the next section. The diverse methods for modeling categorical or limited dependent variables can be accessed under a number of different menus, including

- Statistics > Binary outcomes
- Statistics > Ordinal outcomes
- Statistics > Categorical outcomes
- Statistics > Generalized linear models
- Statistics > Longitudinal/panel data
- Statistics > Linear models and related
- Statistics > Multilevel mixed-effects models

After the Example Commands section, the remainder of this chapter concentrates on an important family of methods called logistic or logit regression. We review basic logit methods for dichotomous, ordinal and multiple-category dependent variables. Chapter 13 introduces a logit version of mixed-effects modeling.

Example Commands

. **logistic y x1 x2 x3**

Performs logistic regression of {0,1} variable y on predictors $x1$, $x2$ and $x3$. Predictor variable effects are reported as odds ratios. A closely related command, **logit**, performs essentially the same analysis, but reports effects as log-odds regression coefficients. The

underlying models fit by **logistic** and **logit** are the same, so predictions and diagnostic tests will be identical.

- **estat gof**

Presents a Pearson chi-squared goodness-of-fit test for the fitted logistic model: observed versus expected frequencies of $y = 1$, using cells defined by the covariate (x -variable) patterns. When a large number of x patterns exist, we might want to group them according to estimated probabilities. **estat gof, group(10)** would perform the test with 10 approximately equal-size groups.

- **estat classification**

Presents classification statistics and classification table. **estat classification, lroc** and **lsens** (see below) are particularly useful when the point of analysis is classification. These commands all refer to the previously-fit logistic model.

- **lroc**

Graphs the receiver operating characteristic (ROC) curve, and calculates area under the curve.

- **lsens**

Graphs both sensitivity and specificity versus the probability cutoff.

- **predict phat**

Generates a new variable (here arbitrarily named *phat*) equal to predicted probabilities that $y = 1$ according to the most recent **logistic** model.

- **predict dx2, dx2**

Generates a new variable (arbitrarily named *dX2*), the diagnostic statistic measuring change in Pearson chi-squared, from the most recent **logistic** analysis.

- **mlogit y x1 x2 x3, base(3) rrr nolog**

Performs multinomial logistic regression of multiple-outcome variable y on three x variables. Option **base(3)** specifies $y = 3$ as the base category for comparison; **rrr** calls for relative risk ratios instead of regression coefficients; and **nolog** suppresses display of the log likelihood on each iteration.

- **svy: mlogit y x1 x2 x3, base(3) rrr nolog**

Survey-weighted multinomial logit regression, requiring that the data have previously been **svyset** (see Chapter 4). Survey versions of **logit**, **ologit** and other modeling commands exist as well, with syntax similar to their non-survey counterparts.

- **predict P2, outcome(2)**

Generates a new variable (arbitrarily named *P2*) representing the predicted probability that $y = 2$, based on the most recent **mlogit** analysis.

- **glm success x1 x2 x3, family(binomial trials) eform**

Performs a logistic regression via generalized linear modeling using tabulated rather than individual-observation data. The variable *success* gives the number of times that the

outcome of interest occurred, and *trials* gives the number of times it could have occurred for each combination of the predictors *x1*, *x2* and *x3*. That is, *success/trials* would equal the proportion of times that an outcome such as “patient recovers” occurred. The **eform** option asks for results in the form of odds ratios (“exponentiated form”) rather than logit coefficients.

Space Shuttle Data

Our first example for this chapter, *shuttle.dta*, involves historical data covering the first 25 flights of the U.S. space shuttle. These data contain evidence that, if properly analyzed, might have persuaded NASA officials not to launch *Challenger* on its fatal flight in 1985 (the 25th shuttle flight, designated STS 51-L). The data are drawn from the *Report of the Presidential Commission on the Space Shuttle Challenger Accident* (1986) and from Tufte (1997). Tufte’s book contains an excellent discussion about data and analytical issues. His comments regarding specific shuttle flights are included as a string variable in these data.

```
. use C:\data\shuttle.dta, clear
. describe

Contains data from C:\data\shuttle.dta
    obs:                  25
    vars:                  8
    size:           1,575
                                         First 25 space shuttle flights
                                         29 May 2012 08:29

variable   storage      display      value
name       type        format     label
variable   label

flight      byte        %8.0g      f1bl      Flight
month       byte        %8.0g      f1bl      Month of launch
day         byte        %8.0g      f1bl      Day of launch
year         int         %8.0g      f1bl      Year of launch
distress    byte        %8.0g      d1bl      Thermal distress incidents
temp         byte        %8.0g      f1bl      Joint temperature, degrees F
damage      byte        %9.0g      f1bl      Damage severity index, Tufte 1997
comments    str55      %55s      f1bl      Comments, Tufte 1997

Sorted by: flight

. list flight-temp, sepby(year)
```

	flight	month	day	year	distress	temp
1.	STS-1	4	12	1981	none	66
2.	STS-2	11	12	1981	1 or 2	70
3.	STS-3	3	22	1982	none	69
4.	STS-4	6	27	1982	.	80
5.	STS-5	11	11	1982	none	68
6.	STS-6	4	4	1983	1 or 2	67
7.	STS-7	6	18	1983	none	72
8.	STS-8	8	30	1983	none	73
9.	STS-9	11	28	1983	none	70
10.	STS_41-B	2	3	1984	1 or 2	57
11.	STS_41-C	4	6	1984	3 plus	63
12.	STS_41-D	8	30	1984	3 plus	70
13.	STS_41-G	10	5	1984	none	78
14.	STS_51-A	11	8	1984	none	67

15.	STS_51-C	1	24	1985	3 plus	53
16.	STS_51-D	4	12	1985	3 plus	67
17.	STS_51-B	4	29	1985	3 plus	75
18.	STS_51-G	6	17	1985	3 plus	70
19.	STS_51-F	7	29	1985	1 or 2	81
20.	STS_51-I	8	27	1985	1 or 2	76
21.	STS_51-J	10	3	1985	none	79
22.	STS_61-A	10	30	1985	3 plus	75
23.	STS_61-B	11	26	1985	1 or 2	76
24.	STS_61-C	1	12	1986	3 plus	58
25.	STS_51-L	1	28	1986	.	31

This chapter examines three of the *shuttle.dta* variables:

distress The number of “thermal distress incidents,” in which hot gas blow-through or charring damaged joint seals of a flight’s booster rockets. Burn-through of a booster joint seal precipitated the *Challenger* disaster. Many previous flights had experienced less severe damage, so the joint seals were known to be a source of possible danger.

temp The calculated joint temperature at launch time, in degrees Fahrenheit. Temperature depends largely on weather. Rubber O-rings sealing the booster rocket joints become less flexible when cold.

date Date, measured in days elapsed since January 1, 1960. *date* is generated from the month, day and year of launch using the **mdy** function (month-day-year to elapsed time; see **help dates**):

```
. generate date = mdy(month, day, year)
. format %td date
. label variable date "Date (days since 1/1/60)"
```

Elapsed date matters because several changes over the course of the shuttle program might have made it riskier. Booster rocket walls were thinned to save weight and increase payloads, and joint seals were subjected to higher-pressure testing. Furthermore, the reusable shuttle hardware was aging. So we might ask, did the probability of booster joint damage (one or more distress incidents) increase with launch date?

distress is a labeled numeric variable:

```
. tabulate distress
```

Thermal distress incidents	Freq.	Percent	Cum.
none	9	39.13	39.13
1 or 2	6	26.09	65.22
3 plus	8	34.78	100.00
Total	23	100.00	

Ordinarily, **tabulate** displays value labels. The **nolabel** option reveals that the numeric codes behind these value labels are 0 = "none", 1 = "1 or 2", and 2 = "3 plus".

```
. tabulate distress, nolabel
```

Thermal distress incidents	Freq.	Percent	Cum.
0	9	39.13	39.13
1	6	26.09	65.22
2	8	34.78	100.00
Total	23	100.00	

We can use these codes to create a new dummy variable, *any*, coded 0 for no distress and 1 for one or more distress incidents:

```
. generate any = distress
. replace any = 1 if distress == 2
. label variable any "Any thermal distress"
```

To check what these **generate** and **replace** commands accomplished, and be sure that missing values were handled correctly,

```
. tabulate distress any, miss
```

Thermal distress incidents	Any thermal distress			Total
	0	1	.	
none	9	0	0	9
1 or 2	0	6	0	6
3 plus	0	8	0	8
.	0	0	2	2
Total	9	14	2	25

Logistic regression models how a {0,1} dichotomy such as *any* depends on one or more *x* variables. The syntax of **logit** resembles that of **regress** and most other model-fitting commands, with the dependent variable listed first.

```
. logit any date
```

```
Iteration 0:  log likelihood = -15.394543
Iteration 1:  log likelihood = -12.997472
Iteration 2:  log likelihood = -12.991097
Iteration 3:  log likelihood = -12.991096
```

```
Logistic regression
```

Number of obs	=	23
LR chi2(1)	=	4.81
Prob > chi2	=	0.0283
Pseudo R2	=	0.1561

```
Log likelihood = -12.991096
```

any	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
date	.0020907	.0010703	1.95	0.051	-6.94e-06 .0041884
_cons	-18.13116	9.517253	-1.91	0.057	-36.78463 .5223142

The **logit** iterative estimation procedure maximizes the log of the likelihood function, shown above the logistic table. At iteration 0, the log likelihood describes the fit of a model including only the constant. The last log likelihood describes the fit of the final model,

$$L = -18.13116 + .0020907date \quad [9.1]$$

where L represents the predicted logit, or log odds, of any distress incidents:

$$L = \ln[P(any = 1) / P(any = 0)] \quad [9.2]$$

An overall χ^2 test at the upper right evaluates the null hypothesis that all coefficients in the model, except the constant, equal zero,

$$\chi^2 = -2(\ln \mathcal{L}_i - \ln \mathcal{L}_f) \quad [9.3]$$

where $\ln \mathcal{L}_i$ is the initial or iteration 0 (model with constant only) log likelihood, and $\ln \mathcal{L}_f$ is the final iteration's log likelihood. Here,

$$\begin{aligned} \chi^2 &= -2[-15.394543 - (-12.991096)] \\ &= 4.81 \end{aligned}$$

The probability of a greater χ^2 , with 1 degree of freedom (the difference in complexity between initial and final models), is low enough (.0283) to reject the null hypothesis in this example. Consequently, *date* does have a significant effect.

Less accurate, though convenient, tests are provided by the asymptotic z (standard normal) statistics displayed with **logit** results. With one predictor variable, that predictor's z statistic and the overall χ^2 statistic test equivalent hypotheses, analogous to the usual t and F statistics in simple OLS regression. Unlike their OLS counterparts, the logit z approximation and χ^2 tests sometimes disagree (they do here). The χ^2 test has more general validity.

Like some other Stata maximum-likelihood procedures, **logit** displays a pseudo R^2 :

$$\text{pseudo } R^2 = 1 - \ln \mathcal{L}_f / \ln \mathcal{L}_i \quad [9.4]$$

For this example,

$$\begin{aligned} \text{pseudo } R^2 &= 1 - (-12.991096) / (-15.394543) \\ &= .1561 \end{aligned}$$

Although pseudo R^2 statistics provide a quick way to describe or compare the fit of different models for the same dependent variable, they lack the straightforward explained-variance interpretation of true R^2 in OLS regression.

After **logit**, the **predict** command (with no options) obtains predicted probabilities,

$$P_{hat} = 1 / (1 + e^{-L}) \quad [9.5]$$

Graphed against *date*, these predicted probabilities follow an S-shaped logistic curve. Because we specified **format %td date** earlier after we defined variable *date*, values are appropriately labeled on the horizontal or time axis in Figure 9.1.

```
. predict Phat
. label variable Phat "Predicted P(distress >= 1)"
. graph twoway connect Phat date, xtitle("Launch date") sort
```

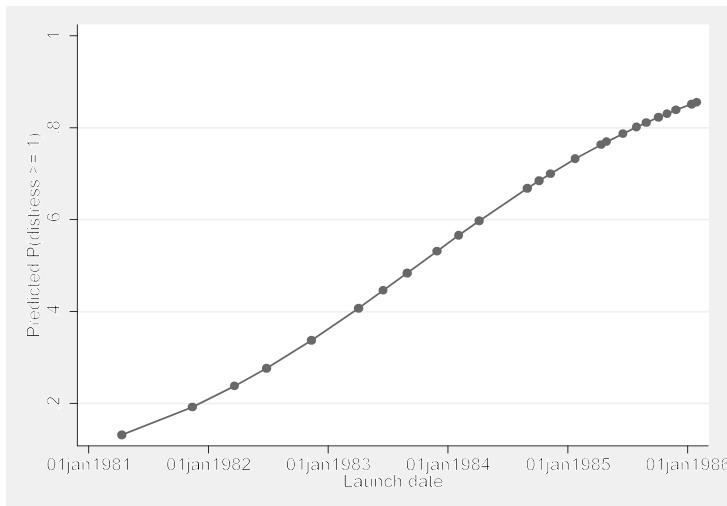


Figure 9.1

The coefficient in this **logit** example (.0020907) describes *date*'s effect on the logit or log odds that any thermal distress incidents occur. Each additional day increases the predicted log odds of thermal distress by .0020907. Equivalently, we could say that each additional day multiplies predicted odds of thermal distress by $e^{.0020907} = 1.0020929$; each 100 days therefore multiplies the odds by $(e^{.0020907})^{100} = 1.23$. ($e \approx 2.71828$, the base number for natural logarithms.) Stata can make these calculations utilizing the `_b[varname]` coefficients stored after any estimation:

```
. display exp(_b[date])
1.0020929

. display exp(_b[date])^100
1.2325358
```

We could also simply include an **or** (odds ratio) option on the **logit** command line. A third way to obtain odds ratios employs the **logistic** command, described in the next section. **logistic** fits exactly the same model as **logit**, but its default output table displays odds ratios rather than coefficients.

Using Logistic Regression

Here is the same regression seen earlier, but using **logistic** instead of **logit**:

```
. logistic any date
```

Logistic regression	Number of obs	=	23			
	LR chi2(1)	=	4.81			
	Prob > chi2	=	0.0283			
	Pseudo R2	=	0.1561			
Log likelihood = -12.991096						
	Odds Ratio	Std. Err.	z	P> z 	[95% Conf. Interval]	
date	1.002093	.0010725	1.95	0.051	.9999931	1.004197
_cons	1.34e-08	1.27e-07	-1.91	0.057	1.06e-16	1.685925

Note the identical log likelihoods and χ^2 statistics. Instead of coefficients (b), **logistic** displays odds ratios (e^b). The numbers in the Odds Ratio column of the **logistic** output are amounts by which the odds favoring $y = 1$ are multiplied, with each 1-unit increase in that x variable (if other x variables' values stay the same).

After fitting a model, we can obtain a classification table and related statistics by typing

estat class						
Logistic model for any						
Classified	True		Total			
	D	~D		Pr(+ D)	85.71%	
+	12	4	16			
-	2	5	7			
Total	14	9	23			
Classified + if predicted Pr(D) >= .5						
True D defined as any != 0						
Sensitivity				Pr(+ D)	85.71%	
Specificity				Pr(- ~D)	55.56%	
Positive predictive value				Pr(D +)	75.00%	
Negative predictive value				Pr(~D -)	71.43%	
False + rate for true ~D				Pr(+ ~D)	44.44%	
False - rate for true D				Pr(- D)	14.29%	
False + rate for classified +				Pr(~D +)	25.00%	
False - rate for classified -				Pr(D -)	28.57%	
Correctly classified					73.91%	

By default, **estat class** employs a probability of .5 as its cutoff (although we can change this by adding a **cutoff()** option). Symbols in the classification table have the following meanings:

- D The event of interest did occur (that is, $y = 1$) for that observation. In this example, D indicates that thermal distress occurred.
- \sim D The event of interest did not occur (that is, $y = 0$) for that observation. In this example, \sim D corresponds to flights having no thermal distress.
- +
- The model's predicted probability is greater than or equal to the cutoff point. Since we used the default cutoff, + here indicates that the model predicts a .5 or higher probability of thermal distress.

- The predicted probability is less than the cutoff. Here, – means a predicted probability of thermal distress below .5.

Thus for 12 flights, classifications are accurate in the sense that the model estimated at least a .5 probability of thermal distress, and distress did in fact occur. For 5 other flights, the model predicted less than a .5 probability, and distress did not occur. The overall correctly-classified rate is therefore $12 + 5 = 17$ out of 23, or 73.91%. The table also gives conditional probabilities such as sensitivity or the percentage of observations with $P \geq .5$ given that thermal distress occurred (12 out of 14 or 85.71%).

After **logistic** or **logit**, the postestimation command **predict** calculates various prediction and diagnostic statistics. Explanations for logit model diagnostic statistics can be found in Hosmer and Lemeshow (2000).

predict newvar	Predicted probability that $y = 1$
predict newvar, xb	Linear prediction (predicted log odds that $y = 1$)
predict newvar, stdp	Standard error of the linear prediction
predict newvar, dbeta	ΔB influence statistic, analogous to Cook's D
predict newvar, deviance	Deviance residual for j th x pattern, d_j
predict newvar, dx2	Change in Pearson χ^2 , written as $\Delta\chi^2$ or $\Delta\chi^2_p$
predict newvar, ddeviance	Change in deviance χ^2 , written as ΔD or $\Delta\chi^2_d$
predict newvar, hat	Leverage of the j th x pattern, h_j
predict newvar, number	Assigns numbers to x patterns, $j = 1, 2, 3 \dots J$
predict newvar, resid	Pearson residual for j th x pattern, r_j
predict newvar, rstandard	Standardized Pearson residual
predict newvar, score	1st derivative of log likelihood with respect to Xb .

Statistics obtained by the **dbeta**, **dx2**, **ddeviance** and **hat** options do not measure the influence of individual observations, as do their counterparts in ordinary regression. Rather, these statistics measure the influence of covariate patterns; that is, the consequences of dropping all observations with that particular combination of x values. See Hosmer and Lemeshow (2000).

Does booster joint temperature also affect the probability of any distress incidents? We could investigate by including *temp* as a second predictor variable.

```
. logistic any date temp
```

Logistic regression	Number of obs = 23
Log Likelihood = -11.350748	LR chi2(2) = 8.09
	Prob > chi2 = 0.0175
	Pseudo R2 = 0.2627

any	Odds Ratio	Std. Err.	z	P> z	[95% Conf. Interval]
date	1.00297	.0013675	2.17	0.030	1.000293 1.005653
temp	.8408309	.0987887	-1.48	0.140	.6678848 1.058561
_cons	1.19e-06	.0000121	-1.34	0.182	2.40e-15 587.9723

Including temperature as a predictor slightly improves the correct classification rate, to 78.26%.

```
. estat class
```

Logistic model for any

classified	True		Total
	D	~D	
+	12	3	15
-	2	6	8
Total	14	9	23

Classified + if predicted Pr(D) >= .5		
True D defined as any != 0		
Sensitivity	Pr(+ D)	85.71%
Specificity	Pr(- ~D)	66.67%
Positive predictive value	Pr(D +)	80.00%
Negative predictive value	Pr(~D -)	75.00%
False + rate for true ~D	Pr(+ ~D)	33.33%
False - rate for true D	Pr(- D)	14.29%
False + rate for classified +	Pr(~D +)	20.00%
False - rate for classified -	Pr(D -)	25.00%
Correctly classified		78.26%

According to the fitted model, each 1-degree increase in joint temperature multiplies the odds of booster joint damage by .84. In other words, each 1-degree warming reduces the odds of damage by about 16%. Although this effect seems strong enough to cause concern, the asymptotic z test says that it is not statistically significant ($z = -1.476$, $p = .140$). A more definitive test, however, employs the likelihood-ratio χ^2 . The **lrtest** command compares nested models estimated by maximum likelihood. First, estimate a full model containing all variables of interest, as done above with the **logistic any date temp** command. Next, type an **estimates store** command, giving a name (such as *full*) to identify this first model:

```
. estimates store full
```

Now estimate a reduced model, including only a subset of the x variables from the full model. (Such reduced models are said to be “nested.”) Finally, a command such as **lrtest full** requests a test of the nested model against the previously stored *full* model. For example (using the **quietly** prefix, because we already saw this output once),

```
. quietly logistic any date
. lrtest full
```

Likelihood-ratio test
(Assumption: *_* nested in *full*)

LR chi2(1) = 3.28
Prob > chi2 = 0.0701

This **lrtest** command tests the recent (presumably nested) model against the model previously saved by **estimates store**. It employs a general test statistic for nested maximum-likelihood models,

$$\chi^2 = -2(\ln \mathcal{L}_1 - \ln \mathcal{L}_0) \quad [9.6]$$

where $\ln \mathcal{L}_0$ is the log likelihood for the first model (with all x variables), and $\ln \mathcal{L}_1$ is the log likelihood for the second model (with a subset of those x variables). Compare the resulting test statistic to a χ^2 distribution with degrees of freedom equal to the difference in complexity (number of x variables dropped) between models 0 and 1. Type **help lrtest** for more about this

command, which works with any of Stata's maximum-likelihood estimation procedures (**logit**, **mlogit**, **stcox** and many others). The overall χ^2 statistic routinely given by **logit** or **logistic** output (equation [9.3]) is a special case of [9.6].

The previous **Irtest** example performed this calculation:

$$\begin{aligned}\chi^2 &= -2[-12.991096 - (-11.350748)] \\ &= 3.28\end{aligned}$$

with 1 degree of freedom, yielding $p = .0701$; the effect of *temp* is significant at $\alpha = .10$. Given the small sample and the disastrous consequences of a Type II error regarding space shuttle safety, $\alpha = .10$ seems a more prudent cutoff than the usual $\alpha = .05$.

Marginal or Conditional Effects Plots

Plotting the adjusted marginal or conditional effects helps to understand and communicate what a logistic model implies about probabilities. For example, we could find the predicted probability of any thermal distress incidents as a function of *temp*, holding *date* constant at relatively low (early) and high (late) values.

. summarize date temp

Variable	Obs	Mean	Std. Dev.	Min	Max
date	25	8905.88	517.6033	7772	9524
temp	25	68.44	10.52806	31	81

Elapsed dates in these data range from the first shuttle flight on April 12, 1981 (*date* = 7772) to the fatal Challenger launch on January 21, 1986 (*date* = 9524). Joint temperatures range from 31 to 81 degrees Fahrenheit. **margins** can calculate and **marginsplot** graph the predicted probabilities from our **logistic** model at 10-degree increments in temperature, at the earliest and latest dates.

```
. quietly logistic any date temp
. margins, at(temp = (30(10)80) date = (7772 9524)) vsquish
```

Adjusted predictions	Number of obs = 23
Model VCE : OIM	
Expression : Pr(any), predict()	
1._at : date = 7772	
: temp = 30	
2._at : date = 7772	
: temp = 40	
3._at : date = 7772	
: temp = 50	
4._at : date = 7772	
: temp = 60	
5._at : date = 7772	
: temp = 70	
6._at : date = 7772	
: temp = 80	
7._at : date = 9524	
: temp = 30	
8._at : date = 9524	
: temp = 40	

```

9._at      : date      =      9524
          temp      =       50
10._at     : date      =      9524
          temp      =       60
11._at     : date      =      9524
          temp      =       70
12._at     : date      =      9524
          temp      =       80

```

	Delta-method				
	Margin	Std. Err.	z	P> z	[95% Conf. Interval]
_at					
1	.985239	.0624661	15.77	0.000	.8628077 1.10767
2	.9218137	.2310152	3.99	0.000	.4690321 1.374595
3	.6755951	.4831333	1.40	0.162	-.2713288 1.622519
4	.2689325	.291999	0.92	0.357	-.3033749 .84124
5	.0610143	.0871295	0.70	0.484	-.1097563 .2317849
6	.0113476	.0255353	0.44	0.657	-.0387006 .0613958
7	.9999169	.0004545	2200.25	0.000	.9990262 1.000808
8	.99953	.0020277	492.94	0.000	.9955558 1.003504
9	.9973449	.0084046	118.67	0.000	.9808722 1.013818
10	.9851528	.0302581	32.56	0.000	.9258479 1.044458
11	.9213867	.0808455	11.40	0.000	.7629324 1.079841
12	.6742985	.2166156	3.11	0.002	.2497398 1.098857

. marginsplot

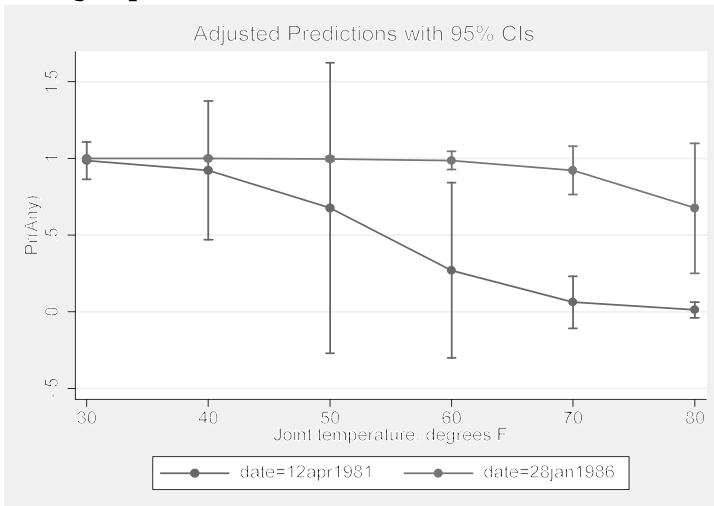


Figure 9.2

The default **marginsplot** graph in Figure 9.2 conveys basic information, but is not pretty. For a more publishable version we could suppress the confidence intervals (**noci**), relocate the legend, use **plot#opts()** to visually distinguish the two curves, and add a title. First, we run **margins** again with one-degree increments in temperature, which will make the resulting curves smoother.

```

. quietly margins, at(temp = (30(1)80) date = (8000 9500))
. marginsplot, noci legend(position(7) ring(0) rows(2))

```

```

plot1opts(msymbol(i) lpattern(dash) lwidth(medthick))
plot2opts(msymbol(i) lpattern(solid) lwidth(medthick))
title("Predicted probability of booster O-ring damage")

```

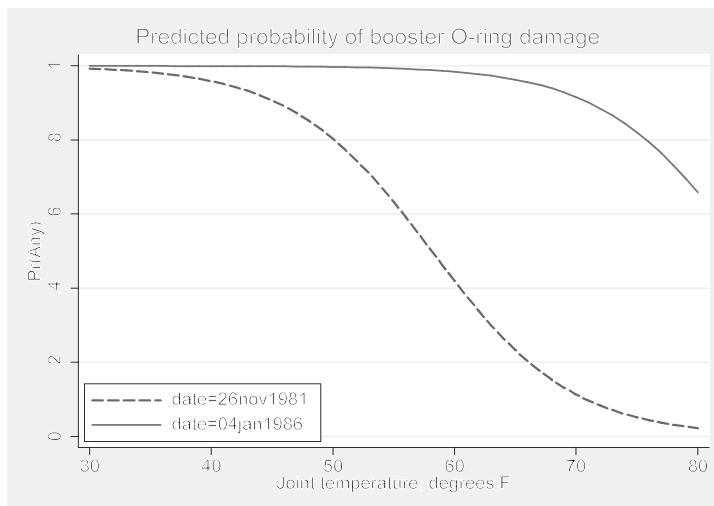


Figure 9.3

According to our logistic model, around the time of the earliest shuttle flight (dashed curve in Figure 9.3) the probability of thermal distress goes from near zero at 80 °F, to near one below 40 °F. By the time of the Challenger flight (solid curve), however, the probability of any distress exceeds .6 even in warm weather, and climbs toward one on flights below 70 °F. Note that *Challenger*'s actual temperature at launch, 31 °F, would place it at top left in Figure 9.3.

Diagnostic Statistics and Plots

As mentioned earlier, the logistic regression influence and diagnostic statistics obtained by **predict** refer not to individual observations, as do the OLS regression diagnostics of Chapter 7. Rather, logistic diagnostics refer to x patterns. In the space shuttle data, however, each x pattern is unique — no two flights share the same combination of *date* and *temp* (naturally, because no two were launched the same day). Before using **predict**, we quietly refit the recent model:

```

. quietly logistic any date temp
. predict Phat3
. label variable Phat3 "Predicted probability"
. predict dx2, dx2
. label variable dx2 "Change in Pearson chi-squared"
. predict dB, dbeta
. label variable dB "Influence"
. predict dD, ddeviance
. label variable dD "Change in deviance"

```

Hosmer and Lemeshow (2000) suggest plots that help in reading these diagnostics. To graph change in Pearson χ^2 versus probability of distress (Figure 9.4), type:

```
. graph twoway scatter dx2 Phat3
```

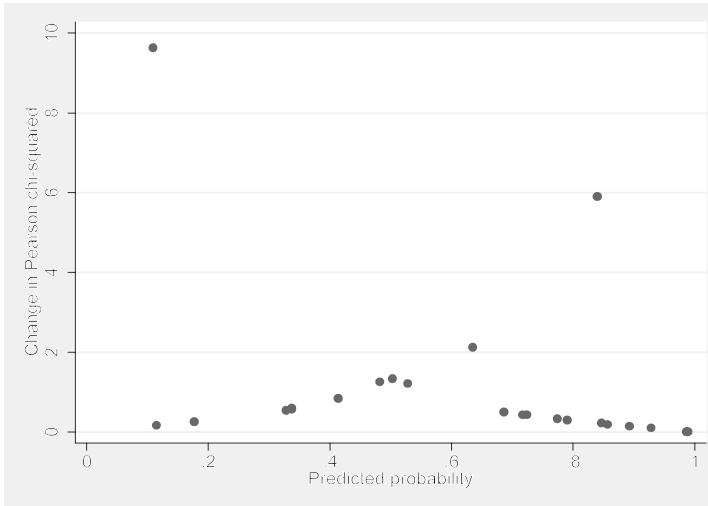


Figure 9.4

Two poorly fit x patterns, at upper right and left in Figure 9.4, stand out. We can visually identify the flights with high $dx2$ values by adding marker labels (*flight* numbers, in this case) to the scatterplot. In Figure 9.5, only those flights with $dx2 > 2$ have been labeled by adding a second overlaid scatterplot. (If we had instead labeled all of the data points, the bottom of the graph would become an unreadable mess.)

```
. graph twoway scatter dx2 Phat3
    || scatter dx2 Phat3 if dx2 > 2, mlabel(flight)
        mlabsize(medsmall)
    || , legend(off)
```

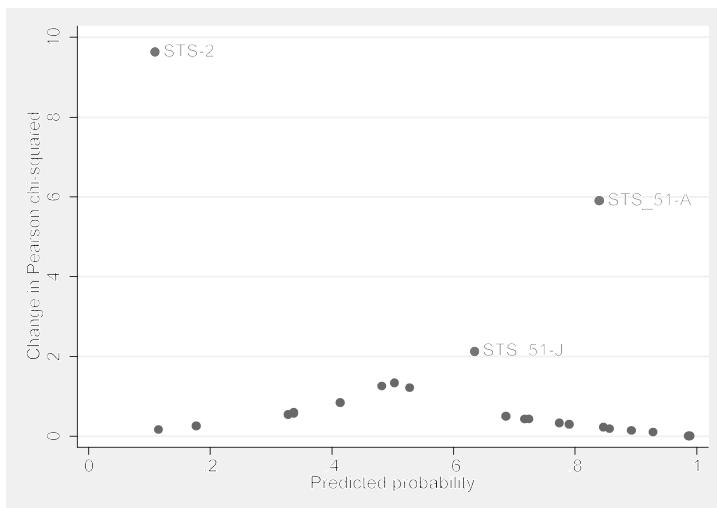


Figure 9.5

```
. list flight any date temp dx2 Phat3 if dx2 > 2
```

	flight	any	date	temp	dx2	Phat3
2.	STS-2	1	12nov1981	70	9.630337	.1091805
4.	STS-4	.	27jun1982	80	.	.0407113
14.	STS_51-A	0	08nov1984	67	5.899742	.8400974
21.	STS_51-J	0	03oct1985	79	2.124642	.6350927
25.	STS_51-L	.	28jan1986	31	.	.9999012

Flight STS 51-A experienced no thermal distress, despite a late launch date and cool temperature (see Figure 9.2). The model predicts a .84 probability of distress for this flight. All points along the up-to-right curve in Figure 9.5 experienced no thermal distress (*any* = 0). Atop the up-to-left (*any* = 1) curve, flight STS-2 experienced thermal distress despite being one of the earliest flights, and launched in slightly milder weather. The model predicts only a .109 probability of distress. Because Stata views missing values as large numbers, it lists the two missing-values flights, including *Challenger*, among those with *dx2* > 2.

Similar findings result from plotting *dd* versus predicted probability, as seen in Figure 9.6. Again, flights STS-2 (top left) and STS 51-A (top right) stand out as poorly fit. Figure 9.6 illustrates a variation on the labeled-marker scatterplot. Instead of putting the flight-number labels near the markers, as done in Figure 9.5 above, we make the markers themselves invisible, **msymbol(i)**, and place labels where the markers would have been, **mlabposition(0)**, for every data point in Figure 9.6.

```
. graph twoway scatter dD Phat3, msymbol(i) mlabposition(0)
    mlabel(flight) mlabsize(small)
```

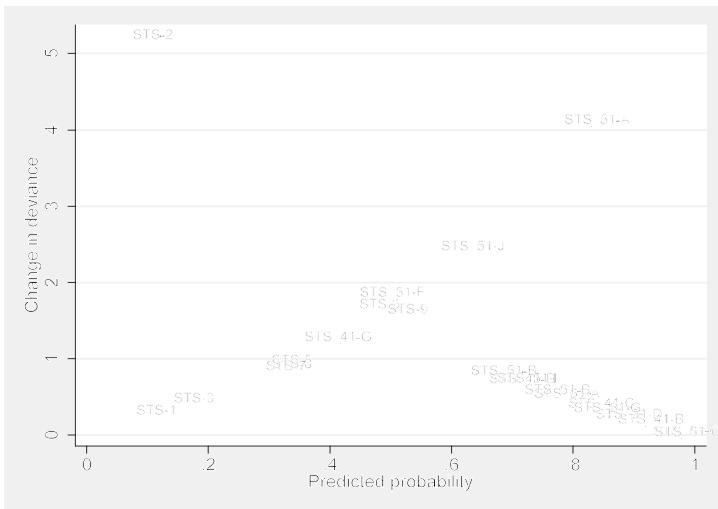


Figure 9.6

dB measures an x pattern's influence in logistic regression. Figure 9.7 has the same design as Figure 9.6, but with marker symbols proportional to influence. The two worst-fit observations are also the most influential.

```
. graph twoway scatter dD Phat3 [aweight = dB], msymbol(oh)
```

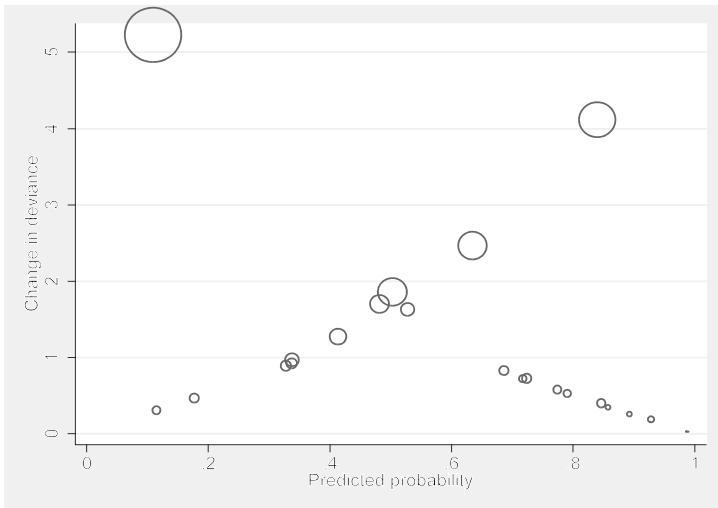


Figure 9.7

Poorly fit and influential observations deserve special attention because they both contradict the main pattern of the data and pull model estimates in their contrary direction. Of course, simply removing such outliers allows a better fit with the remaining data, but this is circular reasoning. A more thoughtful reaction would be to investigate what makes the outliers unusual. Why did shuttle flight STS-2, but not STS 51-A, experience booster joint damage? Seeking an answer might lead investigators to previously overlooked variables.

Logistic Regression with Ordered-Category y

logit and **logistic** fit models for variables with two outcomes, coded 0 and 1. We need other methods for models in which *y* takes on more than two values. Two important possibilities are ordered and multinomial logistic regression.

ologit Ordered logistic regression, where *y* is an ordinal (ordered-category) variable. The numeric values representing the categories do not matter, except that higher numbers mean “more.” For example, the *y* categories might be {1 = “poor,” 2 = “fair,” 3 = “excellent”}.

mlogit Multinomial logistic regression, where *y* has multiple but unordered categories such as {1 = “Democrat” 2 = “Republican,” 3 = “other”}.

If *y* is {0,1}, **logit**, **ologit** and **mlogit** all produce essentially the same estimates.

We earlier simplified the three-outcome ordinal variable *distress* into a dichotomy, *any*. **logit** and **logistic** require {0,1} dependent variables. **ologit**, on the other hand, is designed for ordinal variables that have more than two values. Recall that *distress* has outcomes 0 = “none,” 1 = “1 or 2,” and 2 = “3 plus” incidents of booster-joint distress.

Ordered logistic regression indicates that *date* and *temp* both affect *distress*, with the same signs (positive for *date*, negative for *temp*) seen in our earlier binary logit analyses:

ologit distress date temp, nolog						
Ordered logistic regression						
Log Likelihood = -18.79706						
				Number of obs	=	23
				LR chi2(2)	=	12.32
				Prob > chi2	=	0.0021
				Pseudo R2	=	0.2468
distress	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
date	.003286	.0012662	2.60	0.009	.0008043	.0057677
temp	-.1733752	.0834475	-2.08	0.038	-.3369293	-.0098212
/cut1	16.42813	9.554822			-2.298978	35.15524
/cut2	18.12227	9.722302			-.933092	37.17763

Likelihood-ratio tests are more accurate than the asymptotic *z* tests shown. First, have **estimates store** preserve in memory the results from the full model (with two predictors) just estimated. We can give this model any descriptive name, such as *date_temp*.

```
. estimates store date_temp
```

Next, fit a simpler model without *temp*, store its results with the name *notemp*, and ask for a likelihood-ratio test of whether the fit of reduced model *notemp* differs significantly from that of the full model *date_temp*:

```
. quietly ologit distress date
. estimates store notemp
. lrtest notemp date_temp

Likelihood-ratio test                               LR chi2(1) =      6.12
(Assumption: notemp nested in date_temp)          Prob > chi2 =    0.0133
```

The **lrtest** output notes its assumption that model *notemp* is nested in model *date_temp*, meaning that the parameters estimated in *notemp* are a subset of those in *date_temp*, and that both models are estimated from the same pool of observations (which can be tricky when the data contain missing values). This likelihood-ratio test indicates that *notemp*'s fit is significantly poorer. Because the presence of *temp* as a predictor in model *date_temp* is the only difference, the likelihood-ratio test thus informs us that *temp*'s contribution is significant. Similar steps confirm that *date* also has a significant effect.

```
. quietly ologit distress temp
. estimates store nodate
. lrtest nodate date_temp;

Likelihood-ratio test                               LR chi2(1) =      10.33
(Assumption: nodate nested in date_temp)          Prob > chi2 =   0.0013
```

The **estimates store** and **lrtest** commands provide flexible tools for comparing nested maximum-likelihood models. Type **help lrtest** and **help estimates** for details and options.

The ordered-logit model estimates a score, *S*, for each observation as a linear function of *date* and *temp*:

$$S = .003286date - .1733752temp$$

Predicted probabilities depend on the value of *S*, plus a logistically distributed disturbance *u*, relative to the estimated cut points (shown in **ologit** output as *cut1*, *cut2* etc.).

$$\begin{aligned} P(\text{distress} = \text{"none"}) &= P(S+u \leq \text{cut1}) \\ &= (1 + \exp(-\text{cut1} + S))^{-1} \\ P(\text{distress} = \text{"1 or 2"}) &= P(\text{cut1} < S+u \leq \text{cut2}) \\ &= (1 + \exp(-\text{cut2} + S))^{-1} - (1 + \exp(-\text{cut1} + S))^{-1} \\ P(\text{distress} = \text{"3 plus"}) &= P(\text{cut2} < S+u) \\ &= 1 - (1 + \exp(-\text{cut2} + S))^{-1} \end{aligned}$$

After **ologit**, **predict** calculates predicted probabilities for each category of the dependent variable. We supply **predict** with names for these probabilities. For example: *none* could denote the probability of no distress incidents (first category of *distress*), *onetwo* the probability of 1 or 2 incidents (second category of *distress*), and *threeplus* the probability of 3 or more incidents (third and last category of *distress*):

```
. quietly ologit distress date temp
. predict none onetwo threeplus
```

This creates three new variables:

```
. describe none onetwo threepplus
```

variable name	storage type	display format	value label	variable label
none	float	%9.0g		Pr(distress==0)
onetwo	float	%9.0g		Pr(distress==1)
threepplus	float	%9.0g		Pr(distress==2)

Predicted probabilities for *Challenger*'s last flight, the 25th in these data, are unsettling:

```
. list flight none onetwo threepplus if flight == 25
```

flight	none	onetwo	threep~s
25. STS_51-L	.0000754	.0003346	.99959

Our model, based on the analysis of 23 pre-*Challenger* shuttle flights, predicts little chance ($p = .000075$) of *Challenger* experiencing no booster joint damage, a scarcely greater chance of one or two incidents ($p = .0003$), but virtual certainty ($p = .9996$) of three or more damage incidents.

See Long (1997) or Hosmer and Lemeshow (2000) for detailed presentations of this and related techniques. The *Base Reference Manual* explains Stata's implementation. Long and Freese (2006) provide additional Stata-focused discussion, and make available their ado-files for some useful interpretation and postestimation commands, such as Brant tests. To install these unofficial, free ado-files from the Web, type **findit brant** and follow the link under Web resources.

Multinomial Logistic Regression

When the dependent variable's categories are not ordinal, multinomial logit regression (also called polytomous logit regression) provides appropriate tools. If y has only two categories, **mlogit** (and **ologit**) both fit the same model as **logistic**. Otherwise, though, an **mlogit** model is more complex.

Multiple-category dependent variables occur often in survey data. The Granite State Poll provides a good illustration.

```
. use C:\data\Granite2011_6.dta, clear
. describe age sex educ party warmop2 warmice
```

variable name	storage type	display format	value label	variable label
age	byte	%9.0g	age	Age of respondent
sex	byte	%9.0g	sex2	Gender
educ	byte	%14.0g	educ	Highest degree completed
party	byte	%11.0g	party	Political party identification
warmop2	byte	%9.0g	yesno	Believe happening now/human
warmice	byte	%9.0g	warmice2	Arctic ice vs. 30 years ago

In Chapter 4 we saw that this poll includes three factual questions about climate, such as *warmice*:

- Which of the following three statements do you think is more accurate?*
- Over the past few years, the ice on the Arctic Ocean in late summer ...*
 - covers less area than it did 30 years ago.*
 - declined but then recovered to about the same area it had 30 years ago.*
 - covers more area than it did 30 years ago.*

These data have been **svyset** (see Chapter 4) declaring information about sampling and weights. Commands using the **svy:** prefix will automatically apply this information. For example, weighted response percentages for *warmice* are obtained as follows.

```
. svy: tab warmice, percent
(running tabulate on estimation sample)
```

Number of strata	=	1	Number of obs	=	516
Number of PSUs	=	516	Population size	=	515.57392
			Design df	=	515
<hr/>					
Arctic ice vs. 30 years ago	percentages				
Less	70.91				
Recover	10.43				
More	6.916				
DK/NA	11.75				
Total	100				
Key: percentages = cell percentages					

About 71% of respondents answered correctly that Arctic sea ice area has declined; only 12% said they did not know, or gave no answer.

A second variable of interest indicates whether respondents personally believe that climate change is happening now, caused mainly by human activities (*warmop2*). About 55% believe this is true.

```
. svy: tab warmop2, percent
(running tabulate on estimation sample)
```

Number of strata	=	1	Number of obs	=	516
Number of PSUs	=	516	Population size	=	515.57392
			Design df	=	515
<hr/>					
Believe happening now/human	percentages				
No	45.11				
Yes	54.89				
Total	100				
Key: percentages = cell percentages					

Answers to the *warmice* question correlate with climate-change beliefs, as seen below in a two-way table with percentages based on the row variable, *warmop2*. Accurate answers to *warmice* were given by 83% of those who believe humans are changing the climate, but only 56% of those who do not believe this. The counterfactual response that late-summer Arctic sea ice has recovered to the area it had 30 years ago was four times more common among those who don't think humans are changing the climate. These differences are statistically significant ($p \approx .0000$).

```
. svy: tab warmop2 warmice, row percent
(running tabulate on estimation sample)
```

Number of strata	=	1	Number of obs	=	516
Number of PSUs	=	516	Population size	=	515.57392
			Design df	=	515

Believe happening now/human	Arctic ice vs. 30 years ago				Total
	Less	Recover	More	DK/NA	
No	56.07	17.81	6.951	19.17	100
Yes	83.1	4.354	6.887	5.654	100
Total	70.91	10.43	6.916	11.75	100

Key: row percentages

Pearson:
Uncorrected $\chi^2(3) = 55.2306$
Design-based $F(3.00, 1544.45) = 14.6772 \quad P = 0.0000$

Chapter 4 introduced the **catplot** command, helpful in graphing categorical variables. **catplot** is not supplied with Stata, but can be located by typing **findit catplot**. With **catplot** we can draw a bar chart corresponding to the two-way table above. Treating probability weight variable *censuswt* as analytical weights ([aw=*censuswt*]) results in bars that match the **svy: tab** percentages.

```
. catplot hbar warmice [aw=censuswt], over(warmop2) percent(warmop2)
    blabel(bar, format(%2.0f)) ytitle("Weighted percent")
    title("Arctic sea ice by humans changing climate")
```

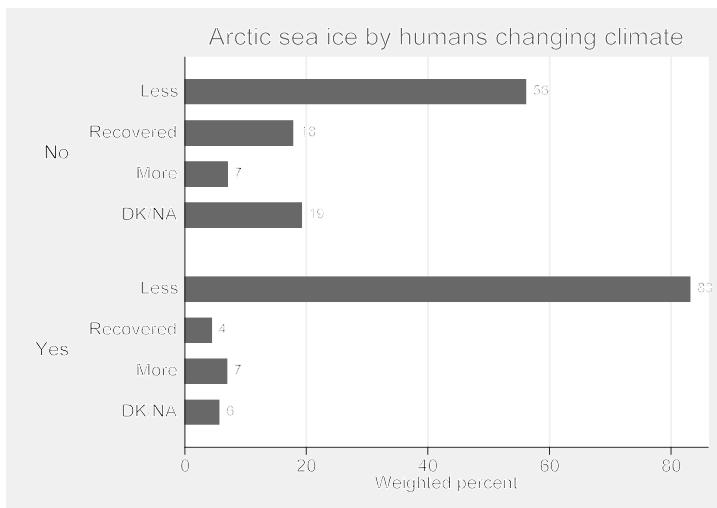
**Figure 9.8**

Figure 9.8, like the percentage table and *F* test, shows a relationship between climate beliefs and facts. You can check for yourself whether similar patterns occur with two other factual questions on this poll, *warmco2* (trends in CO₂) and *warmgre* (greenhouse effect). A traditional explanation for such patterns would be that knowledge informs beliefs, or in this case that knowing specific climate-related facts informs whether people believe humans are changing the climate. Recent social-science research has found evidence of causality in the opposite direction, however. Some people accept specific true or false facts because they fit with their more general beliefs.

In keeping with that hypothesis (termed “biased assimilation”) we could analyze *warmice* responses as a dependent variable. Possible predictors include age, gender, education and political outlook, all of which have often been found to correlate with environmental views. In these poll data, *age* is just age in years; *sex* is coded 0 for males, 1 for females; *educ* ranges from 1 for high school or less to 4 for postgraduate school; and political *party* is coded 1 for Democrats, 2 for Independents and 3 for Republicans.

```
. describe age sex educ party warmop2 warmice
```

variable	name	storage type	display format	value label	variable label
age		byte	%9.0g	age	Age of respondent
sex		byte	%9.0g	sex2	Gender
educ		byte	%14.0g	educ	Highest degree completed
party		byte	%11.0g	party	Political party identification
warmop2		byte	%9.0g	yesno	Believe happening now/human
warmice		byte	%9.0g	warmice2	Arctic ice vs. 30 years ago

How do these background factors affect response to the *warmice* question? Do climate change beliefs still predict responses, once we control for education and politics? **mlogit** results on the next page supply some answers.

```
. svy: mlogit warmice age sex educ party warmop2, rrr base(1)
```

(running mlogit on estimation sample)

Survey: Multinomial logistic regression

Number of strata	=	1	Number of obs	=	486
Number of PSUs	=	486	Population size	=	485.77734
			Design df	=	485
			F(15, 471)	=	4.54
			Prob > F	=	0.0000

warmice	Linearized					
	RRR	Std. Err.	t	P> t	[95% Conf. Interval]	
Less	(base outcome)					
Recovered						
age	1.001732	.0110398	0.16	0.875	.9802738	1.023661
sex	.6975992	.2518093	-1.00	0.319	.3432281	1.417846
educ	.8860304	.1491035	-0.72	0.472	.6365725	1.233245
party	1.718036	.4143614	2.24	0.025	1.069604	2.759569
warmop2	.239992	.1098955	-3.12	0.002	.097599	.590131
_cons	.1196324	.1170444	-2.17	0.030	.0174976	.8179363
More						
age	1.023417	.0147491	1.61	0.109	.9948431	1.052811
sex	.5854667	.2578116	-1.22	0.225	.2464541	1.390812
educ	.5378248	.0936788	-3.56	0.000	.3819503	.7573119
party	1.169189	.3220132	0.57	0.571	.6805561	2.008656
warmop2	1.270082	.6225808	0.49	0.626	.4847726	3.327558
_cons	.0833092	.0846524	-2.45	0.015	.0113137	.6134542
DK_NA						
age	.9866127	.0109802	-1.21	0.226	.9652723	1.008425
sex	1.253388	.4430697	0.64	0.523	.625799	2.51036
educ	.8338215	.1369808	-1.11	0.269	.6037919	1.151487
party	1.707791	.3624779	2.52	0.012	1.125423	2.591516
warmop2	.2443751	.1004212	-3.43	0.001	.1089927	.5479193
_cons	.2678258	.2778029	-1.27	0.205	.0348926	2.055758

This example employs survey weighting. The syntax would be identical (but without the **svy:** prefix) if we had non-survey data. **base(1)** specifies that category 1 (*warmice* = “less area”) should be the base outcome for comparison, so our table shows the predictors of three different wrong answers. The **rrr** option instructs **mlogit** to show relative risk ratios, which resemble the odds ratios given by **logistic**.

In general, the relative risk ratio for outcome j of y , and predictor x_k , equals the amount by which predicted odds favoring $y=j$ (compared with $y=\text{base}$) are multiplied, per 1-unit increase in x_k , other things being equal. In other words, the relative risk ratio rrr_{jk} is a multiplier such that, if all x variables except x_k stay the same,

$$\text{rrr}_{jk} \times \frac{P(y=j | x_k)}{P(y=\text{base} | x_k)} = \frac{P(y=j | x_k+1)}{P(y=\text{base} | x_k+1)}$$

Relative risk ratios in the example above describe the multiplicative effect of a unit increase in each predictor on the odds of selecting a particular *warmice* response instead of the (correct) base category “less area.” We see that Republicans are significantly ($p = .025$) more likely to think ice has recovered, whereas those who believe in human-caused climate change are significantly ($p = .002$) less likely to do so. Other things being equal, the odds of a Republican

responding that Arctic ice has recovered (rather than covering less area) are 72% higher (multiplied by 1.72) compared with those of an Independent, and about 196% higher (multiplied by $1.72^2 = 2.96$) than those of a Democrat. People who believe climate change is happening now due to humans have 76% lower odds (multiplied by 0.24) of saying that ice recovered instead of declined.

The second and third sub-tables in the **mlogit** output give relative risk ratios favoring each of the other *warmice* responses, compared with “less.” The response that Arctic ice covers more area than it did 30 years ago is favored only by less educated respondents. Odds of saying the ice covers more area decrease by 46% (multiplied by 0.54) with each 1-unit increase in *educ*, other things being equal. Thus, the “recovered” response has belief or political predictors, whereas “more” apparently reflects simple lack of knowledge. The “don’t know” or “no answer” (DK/NA) response also has belief and political predictors, perhaps indicating a rejection of the question.

margins and **marginsplot** can visualize our results. The following commands produce a rough plot (Figure 9.9) showing predicted probabilities that *warmice* = “less area,” as a function of climate beliefs (*warmop2*) and political party, based on the previous **mlogit** model. By specifying **predict(outcome(1))** in the **margins** command, we focus on the first outcome of our dependent variable, “less area.”

```
. margins, at(party = (1 2 3) warmop2 = (1 0))
    vsquish predict(outcome(1))

Predictive margins                                         Number of obs   =      486
Model VCE   : Linearized

Expression  : Pr(warmice==Less), predict(outcome(1))
1._at       : party      =      1
              : warmop2   =      1
2._at       : party      =      1
              : warmop2   =      0
3._at       : party      =      2
              : warmop2   =      1
4._at       : party      =      2
              : warmop2   =      0
5._at       : party      =      3
              : warmop2   =      1
6._at       : party      =      3
              : warmop2   =      0
```

	Delta-method					
	Margin	Std. Err.	z	P> z	[95% Conf. Interval]	
_at						
1	.8607879	.0250921	34.31	0.000	.8116084	.9099675
2	.7289992	.0544082	13.40	0.000	.622361	.8356373
3	.8143384	.0286563	28.42	0.000	.7581732	.8705037
4	.626633	.0416184	15.06	0.000	.5450624	.7082036
5	.7500431	.0495879	15.13	0.000	.6528526	.8472336
6	.5072642	.0445321	11.39	0.000	.419983	.5945455

```
. marginsplot
```

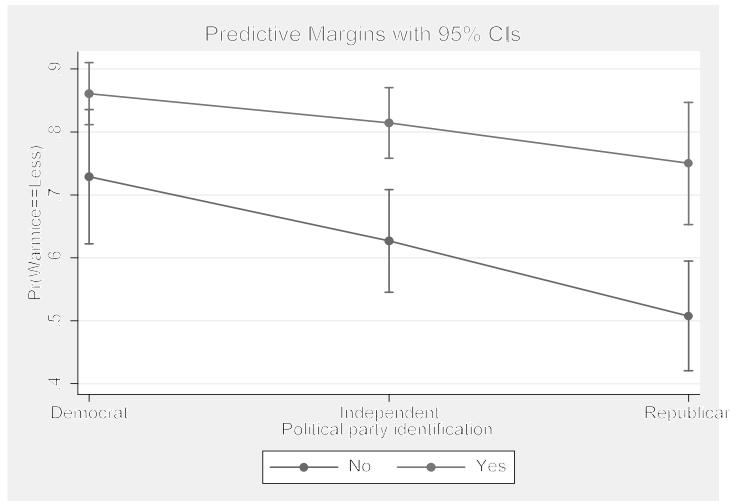


Figure 9.9

To graph the predicted probability for the second outcome, *warmice* = “recovered,” we repeat the **margins** command but with **predict(outcome(2))**. Figure 9.10 shows the result.

```
. margins, at(party = (1 2 3) warmop2 = (1 0))
vsquish predict(outcome(2))
```

		Number of obs = 486			
Predictive margins					
Model VCE : Linearized					
Expression	: Pr(warmice==Recovered), predict(outcome(2))				
1._at	: party = 1	warmop2 = 1			
2._at	: party = 1	warmop2 = 0			
3._at	: party = 2	warmop2 = 1			
4._at	: party = 2	warmop2 = 0			
5._at	: party = 3	warmop2 = 1			
6._at	: party = 3	warmop2 = 0			
		Delta-method			
		Margin	Std. Err.	z	P> z
					[95% Conf. Interval]
<u>_at</u>					
1	.0290269	.0112526	2.58	0.010	.0069723 .0510815
2	.1023611	.0404401	2.53	0.011	.0231001 .1816222
3	.0470891	.0156777	3.00	0.003	.0163614 .0778168
4	.1507864	.0329816	4.57	0.000	.0861437 .2154292
5	.0743527	.0306377	2.43	0.015	.014304 .1344015
6	.2091542	.0372832	5.61	0.000	.1360805 .282228

```
. marginsplot
```

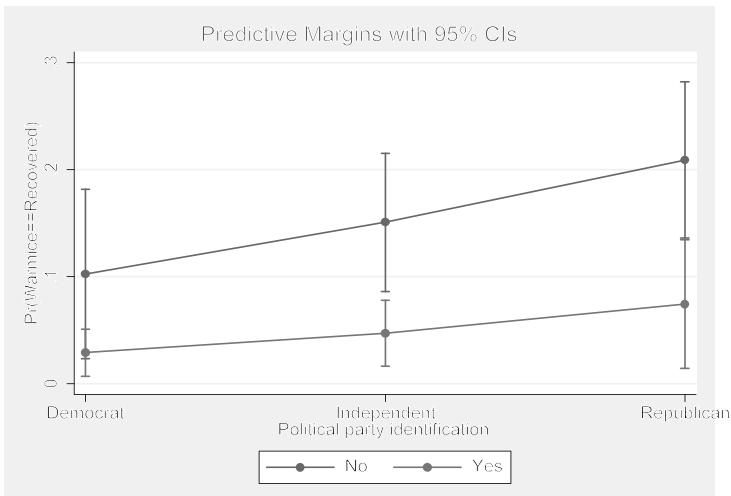


Figure 9.10

Figure 9.11 is a cleaned-up version of Figure 9.9, illustrating the use of some **marginsplot** options. We begin by **quietly** repeating **margins** for outcome 1 (less area), then issue a new **marginsplot** command with options that control details of the labeling, legend and lines. The x-axis range is made a little wider, from 1 to 3.1 instead of the default 1 to 3, in order to accommodate the label “Republican” at lower right.

```
. quietly margins, at(party = (1 2 3) warmop2 = (1 0))
    predict(outcome(1))
. marginsplot, legend(position(7) ring(0) rows(1)
    title("Believe humans" "changing climate?", size(medsmall))
    xscale(range(1 3.1)) ytitle("Probability")
    plot1opts(lpattern(dash) lwidth(medthick) msymbol(O))
    plot2opts(lpattern(solid) lwidth(medthick) msymbol(Sh))
    title("Predicted probability of 'Artic ice area less' response")
```

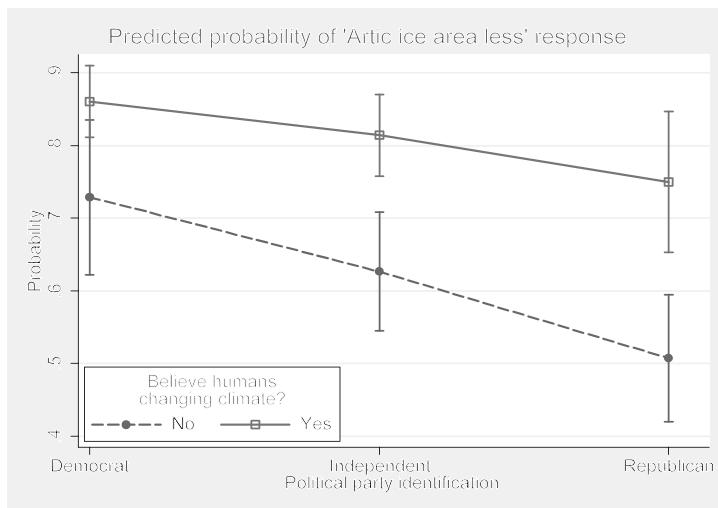


Figure 9.11

Multiple Imputation of Missing Values — Logit Regression Example

Chapter 8 introduced Stata's methods for multiple imputation of missing values, illustrated by a regression example. Multiple-imputation methods work with other types of analysis as well, including the logit-type models discussed in this chapter. For an illustration, we return to the Granite State Poll data and the climate-change belief indicator *warmop2*. The previous section tested age, gender, education and political party as possible predictors of response to climate-knowledge question *warmice*. Those four background characteristics are the usual suspects in research on the social bases of environmental concern, so it is reasonable to guess that one or more will be related to *warmop2*. Should we also consider household income, another important background characteristic, as a possible predictor? One problem with income on surveys is that it tends to have a lot of missing values, because many people feel disinclined to answer this question.

Ten variables from *Granite2011_06.dta* will be used in this analysis. Four of these (*employ*, *ownrent*, *married* and *yrslive*) hold no theoretical interest with respect to climate change beliefs, but might prove helpful for imputing the missing values of *income*.

```
. use C:\data\Granite2011_6.dta, clear
. describe warmop2 age sex educ party income employ ownrent
   married yrslive
```

variable name	storage type	display format	value label	variable label
warmop2	byte	%9.0g	yesno	Believe happening now/human
age	byte	%9.0g	age	Age of respondent
sex	byte	%9.0g	sex2	Gender
educ	byte	%14.0g	educ	Highest degree completed
party	byte	%11.0g	party	Political party identification
income	byte	%9.0g	income3	Household income 2009
employ	byte	%13.0g	employ	Employment status
ownrent	byte	%13.0g	ownrent	Own or rent home
married	byte	%9.0g	yesno	Respondent married
yrslive	byte	%8.0g	yrslive	Years lived in NH

```
. misstable summarize warmop2 age sex educ party income employ
   ownrent married yrslive
```

Variable	Obs=.	Obs>.	Obs<.	obs<.		
				unique values	Min	Max
age	23		493	74	18	94
educ		5	511	4	1	4
party		13	503	3	1	3
income		171	345	7	1	7
employ		16	500	8	1	8
ownrent		20	496	2	0	1
yrslive		12	504	4	1	4

Although we listed *warmop2*, *sex* and *married* in the **misstable** command, Stata detects that they have no missing values and does not include them in the output. On the other hand, out of 516 interviews, we have 171 missing values on *income*. If we regress the dichotomous variable *warmop2* on *income* along with the usual suspects, our estimation sample includes just 340 observations.

```
. svy: logit warmop2 age sex educ party income
(running logit on estimation sample)
```

Survey: Logistic regression

Number of strata = 1	Number of obs = 340
Number of PSUs = 340	Population size = 336.84437
	Design df = 339
	F(5, 335) = 13.47
	Prob > F = 0.0000

warmop2	Coef.	Linearized Std. Err.	t	P> t	[95% Conf. Interval]
age	-.014292	.0092398	-1.55	0.123	-.0324666 .0038826
sex	.4762698	.2829395	1.68	0.093	-.0802685 1.032808
educ	.2508435	.1525283	1.64	0.101	-.0491775 .5508646
party	-1.176907	.1547857	-7.60	0.000	-1.481369 -.8724461
income	.0366035	.0768294	0.48	0.634	-.114519 .1877259
_cons	2.252896	.7778417	2.90	0.004	.7228924 3.7829

Only political party appears to have a significant effect. Would we reach the same conclusion if we could run this analysis without setting so much of the data aside? Multiple imputation provides a way to approach that question.

As a first step for imputation, we drop the 42 observations that have missing values on any of the variables of interest, except for *income*. After doing this we have a dataset with $516 - 42 = 474$ observations, including 137 for which *income* is missing.

```
. keep if !missing(warmop2, age, sex, educ, party, employ,
    ownrent, married, yrslive)
(42 observations deleted)

. misstable summarize warmop2 age sex educ party income employ
    ownrent married yrslive
          obs<.
```

Variable	Obs=.	Obs>.	Obs<.	Unique values	Min	Max
income	137		337	7	1	7

Next we set the multiple imputation data format as **mlong**, a memory-efficient choice. *income* is registered as **imputed**, meaning we will try to fill in its missing values. Other variables are registered as **regular**, so they will not be imputed.

```
. mi set mlong
. mi register imputed income
(137 m=0 obs. now marked as incomplete)

. mi register regular warmop2 sex educ party employ ownrent
    married yrslive
```

137 observations with missing *income* values are predicted by regression on *employ*, *ownrent*, *married* and *yrslive*. Fifty sets of imputed values are created, each with these 137 predicted values plus random noise. The imputations are then pooled to estimate a new logit regression model.

```
. mi impute regress income employ ownrent married yrslive,
    add(50) rseed(12345)
```

Univariate imputation	Imputations =	50
Linear regression	added =	50
Imputed: m=1 through m=50	updated =	0

Variable	Observations per m		
	Complete	Incomplete	Imputed
income	337	137	137

(complete + incomplete = total; imputed is the minimum across m
of the number of filled-in observations.)

```
. mi estimate: svy: logit warmop2 age sex educ party income
```

Multiple-imputation estimates Survey: Logistic regression	Imputations = 50
Number of strata = 1	Number of obs = 474
Number of PSUs = 474	Population size = 472.04182
	Average RVI = 0.0298
	Largest FMI = 0.1511
	Complete DF = 473
DF adjustment: Small sample	DF: min = 338.55
	avg = 443.42
	max = 470.18
Model F test: Equal FMI	F(5, 469.8) = 16.88
Within VCE type: Linearized	Prob > F = 0.0000

warmop2	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
age	-.0188856	.0075324	-2.51	0.013	-.033687 -.0040842
sex	.4338802	.2372722	1.83	0.068	-.0323649 .9001254
educ	.2489546	.1262441	1.97	0.049	.0008665 .4970427
party	-1.154414	.1318129	-8.76	0.000	-1.41343 -.8953988
income	.0134369	.0681875	0.20	0.844	-.1206877 .1475614
_cons	2.669003	.6581604	4.06	0.000	1.375593 3.962413

Following multiple imputation, the logit coefficient on *party* remains similar (-1.15 compared with the previous -1.18), and still statistically significant. Other results show greater change, including shifts in coefficients but also generally smaller standard errors, reflecting more precise estimates from the imputation-enhanced data. Through these changes the coefficients on *age* (negative) and *educ* (positive) now appear statistically significant as well, in keeping with results from many previous studies of climate-change beliefs. *income*, on the other hand, shows little effect in either the pre- or post-imputation model. That finding could support a decision to leave *income* out of a final model, and focus on other more important and less troublesome predictors.

Survival and Event-Count Models

This chapter presents methods for analyzing event data. *Survival analysis* involves several related techniques that focus on times until the event of interest occurs. Although the event could be good or bad, by convention we refer to the event as a “failure.” The time until failure is “survival time.” Survival analysis is important in biomedical research, but it can be applied equally well to other fields from engineering to social science — for example, in modeling the time until an unemployed person gets a job, or a single person gets married. Stata offers a full range of survival analysis procedures, a few of which are illustrated in this chapter.

We also look briefly at Poisson regression and its relatives. These methods focus not on survival times but, rather, on the rates or counts of events over a specified interval of time. Event-count methods include Poisson regression and negative binomial regression. Such models can be fit either through specialized commands or through the broader approach of generalized linear modeling (GLM).

Consult the *Survival Analysis and Epidemiological Tables Reference Manual* for more information about Stata’s capabilities. Type **help st** to see an online overview. Selvin (2004, 2008) provides well-illustrated introductions to survival analysis and Poisson regression. I have borrowed (with permission) several of his examples. Other good introductions to survival analysis include the Stata-oriented volume by Cleves et al. (2010), a chapter in Rosner (1995), and comprehensive treatments by Hosmer, Lemeshow and May (2008) and Lee (1992). McCullagh and Nelder (1989) describe generalized linear models. Long (1997) has a chapter on regression models for count data (including Poisson and negative binomial), and also some material on generalized linear models. An extensive and current treatment of generalized linear models is found in Hardin and Hilbe (2012).

Stata menu groups most relevant to this chapter include:

Statistics > Survival analysis

Graphics > Survival analysis graphs

Statistics > Count outcomes

Statistics > Generalized linear models

Regarding epidemiological tables, not covered in this chapter, further information can be found by typing **help epitab** or exploring the menus for

Statistics > Epidemiology and related

Example Commands

Most of Stata's survival-analysis (**st***) commands require that the data have previously been identified as survival-time by an **stset** command. **stset** need only be run once, and the data subsequently saved.

. stset timevar, failure(failvar)

Identifies single-record survival-time data. Variable *timevar* indicates the time elapsed before either a particular event (called a "failure") occurred, or the period of observation ended ("censoring"). Variable *failvar* indicates whether a failure (*failvar* = 1) or censoring (*failvar* = 0) occurred at *timevar*. The dataset contains only one record per individual. The dataset must be **stset** before any further **st*** commands will work. If we subsequently **save** the dataset, however, the **stset** definitions are saved as well. **stset** creates new variables named *_st*, *_d*, *_t* and *_t0* that encode information necessary for subsequent **st*** commands.

. stset timevar, failure(failvar) id(patient) enter(time start)

Identifies multiple-record survival-time data. In this example, the variable *timevar* indicates elapsed time before failure or censoring; *failvar* indicates whether failure (1) or censoring (0) occurred at this time. *patient* is an identification number. The same individual might contribute more than one record to the data, but always has the same identification number. *start* records the time when each individual came under observation.

. stdescribe

Describes survival-time data, listing definitions set by **stset** and other characteristics of the data.

. stsum

Obtains summary statistics: the total time at risk, incidence rate, number of subjects, and percentiles of survival time.

. ctset time nfail ncensor nenter, by(ethnic sex)

Identifies count-time data. In this example, the variable *time* is a measure of time; *nfail* is the number of failures occurring at *time*. We also specified *ncensor* (number of censored observations at *time*) and *nenter* (number entering at *time*), although these can be optional. *ethnic* and *sex* are other categorical variables defining observations in these data.

. cttost

Converts count-time data, previously identified by the **ctset** command, into survival-time form that can be analyzed by **st*** commands.

- **sts graph**

Graphs the Kaplan–Meier survivor function. To visually compare two or more survivor functions, such as one for each value of the categorical variable *sex*, use a **by()** option such as **sts graph, by(sex)**. To adjust, through Cox regression, for the effects of a continuous independent variable such as *age*, use an **adjustfor()** option such as **sts graph, by(sex) adjustfor(age)**. The **by()** and **adjustfor()** options work similarly with the **sts list** and **sts generate** commands.

- **sts list**

Lists the estimated Kaplan–Meier survivor (or failure) function.

- **sts test sex**

Tests the equality of the Kaplan–Meier survivor function across categories of *sex*.

- **sts generate survfunc = S**

Creates a new variable arbitrarily named *survfunc*, containing the estimated Kaplan–Meier survivor function.

- **stcox x1 x2 x3**

Fits a Cox proportional hazard model, regressing time to failure on continuous or dummy variable predictors *x1*, *x2* and *x3*.

- **stcox x1 x2 x3, strata(x4) vce(robust)**

- **predict hazard, basehazard**

Fits a Cox proportional hazard model, stratified by *x4*. The **vce(robust)** option requests robust standard error estimates. See Chapter 8, or for a more complete explanation of robust standard errors, consult the *User's Guide*. The **predict** command stores the group-specific baseline cumulative hazard function as a new variable named *hazard*; type **help stcox postestimation** for more options.

- **stphplot, by(sex)**

Plots $-\ln(-\ln(\text{survival}))$ versus $\ln(\text{analysis time})$ for each level of the categorical variable *sex*, from the previous **stcox** model. Roughly parallel curves support the Cox model assumption that the hazard ratio does not change with time. Other checks on the Cox assumptions are performed by the commands **stcoxkm** (compares Cox predicted curves with Kaplan–Meier observed survival curves) and **estat phtest** (performs test based on Schoenfeld residuals). See **help stcox diagnostics** for syntax and options.

- **streg x1 x2, dist(weibull)**

Fits Weibull-distribution model regression of time-to-failure on continuous or dummy variable predictors *x1* and *x2*.

- **streg x1 x2 x3 x4, dist(exponential) vce(robust)**

Fits exponential-distribution model regression of time-to-failure on continuous or dummy predictors *x1*–*x4*. Obtains heteroskedasticity-robust standard error estimates. In addition to Weibull and exponential, other **dist()** specifications for **streg** include lognormal, log-logistic, Gompertz or generalized gamma distributions. Type **help streg** for more information.

. stcurve, survival

After **streg**, plots the survival function from this model at mean values of all the *x* variables.

. stcurve, cumhaz at(x3=50, x4=0)

After **streg**, plots the cumulative hazard function from this model at mean values of *x1* and *x2*, *x3* set at 50, and *x4* set at 0.

. poisson count x1 x2 x3, irr exposure(x4)

Performs Poisson regression of event-count variable *count* (assumed to follow a Poisson distribution) on continuous or dummy independent variables *x1*–*x3*. Independent-variable effects will be reported as incidence rate ratios (**irr**). The **exposure()** option identifies a variable indicating the amount of exposure, if this is not the same for all observations.

Note: A Poisson model assumes that the event probability remains constant, regardless of how many times an event occurs for each observation. If the probability does not remain constant, we should consider using **nbreg** (negative binomial regression) or **gnbreg** (generalized negative binomial regression) instead.

. glm count x1 x2 x3, link(log) family(poisson) exposure(x4) eform

Performs the same regression specified in the **poisson** example above, but as a generalized linear model (GLM). **glm** can fit Poisson, negative binomial, logit and many other types of models, depending on what **link()** (link function) and **family()** (distribution family) options we employ.

Survival-Time Data

Survival-time data contain, at a minimum, one variable measuring how much time elapsed before a certain event occurred for each observation. The literature often terms this event of interest a “failure,” regardless of its real-world meaning. When failure has not occurred to an observation by the time data collection ends, that observation is said to be “censored.” The **stset** command sets up a dataset for survival-time analysis by identifying which variable measures time and (if necessary) which variable is a {0, 1} indicator for whether the observation failed or was censored. The dataset can also contain any number of other measurement or categorical variables. Individuals (for example, medical patients) can be represented by more than one observation.

To illustrate the use of **stset**, we will begin with an example from Selvin (1995:453) concerning 51 individuals diagnosed with HIV. The data initially reside in a raw-data file (*aids.raw*) that looks like this:

1	1	1	34
2	17	1	42
3	37	0	47
(rows 4–50 omitted)			
51	81	0	29

The first column values are case numbers (1, 2, 3, . . . , 51). The second column tells how many months elapsed after the diagnosis, before that person either developed symptoms of AIDS or the study ended (1, 17, 37, . . .). The third column holds a 1 if the individual developed AIDS

symptoms (failure), or a 0 if no symptoms had appeared by the end of the study (censoring). The last column reports the individual's age at the time of diagnosis.

We can read the raw data into memory using **infile**, then label the variables and data:

```
. infile case time aids age using C:\data\aids.raw, clear
. label variable case "Case ID number"
. label variable time "Months since HIV diagnosis"
. label variable aids "Developed AIDS symptoms"
. label variable age "Age in years"
. label data "AIDS (Selvin 1995:453)"
. compress
```

The next step is to identify which variable measures time and which indicates failure/censoring. Although not necessary with these single-record data, we can also note which variable holds individual case identification numbers. In an **stset** command, the first-named variable measures time. Subsequently, we identify with **failure()** the dummy representing whether an observation failed (1) or was censored (0). After using **stset**, we save the dataset in Stata format to preserve this information.

```
. stset time, failure(aids) id(case)
      id: case
      failure event: aids != 0 & aids < .
obs. time interval: (time[_n-1], time]
exit on or before: failure

-----
```

51	total obs.
0	exclusions
<hr/>	
51	obs. remaining, representing
51	subjects
25	failures in single failure-per-subject data
3164	total analysis time at risk, at risk from t = 0
	earliest observed entry t = 0
	last observed exit t = 97

```
. save aids.dta, replace
```

stdescribe yields a brief description of how our survival-time data are structured. In this simple example we have only one record per subject, so some of this information is unneeded.

```
. stdescribe
```

failure _d: aids analysis time _t: time id: case					
Category	total	per subject			
		mean	min	median	max
no. of subjects	51				
no. of records	51	1	1	1	1
(first) entry time		0	0	0	0
(final) exit time		62.03922	1	67	97
subjects with gap	0				
time on gap if gap	0				
time at risk	3164	62.03922	1	67	97
failures	25	.4901961	0	0	1

The **stsum** command obtains summary statistics. We have 25 failures out of 3,164 person-months, giving an incidence rate of $25/3164 = .0079014$. The percentiles of survival time derive from a Kaplan–Meier survivor function (next section). This function estimates about a 25% chance of developing AIDS within 41 months after diagnosis, and 50% within 81 months. Over the observed range of the data (up to 97 months) the probability of AIDS does not reach 75%, so there is no 75th percentile given.

. **stsum**

failure _d: aids analysis time _t: time id: case						
	time at risk	incidence rate	no. of subjects	Survival time		
				25%	50%	75%
total	3164	.0079014	51	41	81	.

If the data happen to include a grouping or categorical variable such as *sex*, we could obtain summary statistics on survival time separately for each group by a command of the following form:

. **stsum, by(sex)**

Later sections describe more formal methods for comparing survival times from two or more groups.

Count-Time Data

Survival-time (**st**) datasets like *aids.dta* contain information on individual people or things, with variables indicating the time at which failure or censoring occurred for each individual. A different type of dataset called count-time (**ct**) contains aggregate data, with variables counting the number of individuals that failed or were censored at time *t*. For example, *diskdriv.dta* contains hypothetical test information on 25 disk drives. All but 5 drives failed before testing ended at 1,200 hours.

```
. use C:\data\diskdriv.dta, clear
. describe

Contains data from C:\data\diskdriv.dta
obs: 6
vars: 3
size: 24
                                         Count-time data on disk drives
                                         30 Jun 2012 10:19

variable   storage   display   value
name      type      format    label
hours     int       %8.0g
failures byte      %8.0g
censored  byte      %9.0g
                                         Hours of continuous operation
                                         Number of failures observed
                                         Number still working

Sorted by:

. list



|    | hours | failures | censored |
|----|-------|----------|----------|
| 1. | 200   | 2        | 0        |
| 2. | 400   | 3        | 0        |
| 3. | 600   | 4        | 0        |
| 4. | 800   | 8        | 0        |
| 5. | 1000  | 3        | 0        |
| 6. | 1200  | 0        | 5        |


```

To set up a count-time dataset, we specify the time variable, the number-of-failures variable, and the number-censored variable, in that order. After **ctset**, the **cttost** command automatically converts our count-time data to survival-time format.

```
. ctset hours failures censored

dataset name: C:\data\diskdriv.dta
      time: hours
      no. fail: failures
      no. lost: censored
      no. enter: --          (meaning all enter at time 0)

. cttost

      failure event: failures != 0 & failures < .
obs. time interval: (0, hours]
exit on or before: failure
      weight: [fweight=w]



|   |            |
|---|------------|
| 6 | total obs. |
| 0 | exclusions |



|       |                                                 |
|-------|-------------------------------------------------|
| 6     | physical obs. remaining, equal to               |
| 25    | weighted obs., representing                     |
| 20    | failures in single record/single failure data   |
| 19400 | total analysis time at risk, at risk from t = 0 |
|       | earliest observed entry t = 0                   |
|       | last observed exit t = 1200                     |



. list
```

	hours	failures	censored	w	_st	_d	_t	_t0
1.	200	1	0	2	1	1	200	0
2.	400	1	0	3	1	1	400	0
3.	600	1	0	4	1	1	600	0
4.	800	1	0	8	1	1	800	0
5.	1000	1	0	3	1	1	1000	0
6.	1200	0	5	5	1	0	1200	0

. stdescribe

```
failure _d: failures
analysis time _t: hours
weight: [fweight=w]
```

Category	unweighted total	unweighted mean	per subject		
			min	unweighted median	max
no. of subjects	6				
no. of records	6	1	1	1	1
(first) entry time		0	0	0	0
(final) exit time		700	200	700	1200
subjects with gap	0				
time on gap if gap	0				
time at risk	4200	700	200	700	1200
failures	5	.8333333	0	1	1

The **cttost** command defines a set of frequency weights, *w*, in the resulting **st**-format dataset. **st*** commands automatically recognize and use these weights in any survival-time analysis, so the data now are viewed as containing 25 observations (25 disk drives) instead of the previous 6 (six time periods).

. stsum

```
failure _d: failures
analysis time _t: hours
weight: [fweight=w]
```

	time at risk	incidence rate	no. of subjects	Survival time		
				25%	50%	75%
total	19400	.0010309	25	600	800	1000

Kaplan–Meier Survivor Functions

Let n_t represent the number of observations that have not failed, and are not censored, at the beginning of time period t . d_t represents the number of failures that occur to these observations during time period t . The Kaplan–Meier estimator of surviving beyond time t is the product of survival probabilities in t and the preceding periods:

$$S(t) = \prod_{j=0}^t \left\{ (n_j - d_j) / n_j \right\} \quad [10.1]$$

For example, in the AIDS data seen earlier, one of the 51 individuals developed symptoms only one month after diagnosis. No observations were censored this early, so the probability of “surviving” (meaning, not developing AIDS) beyond $time = 1$ is

$$S(1) = (51 - 1) / 51 = .9804$$

A second patient developed symptoms at $time = 2$, and a third at $time = 9$:

$$S(2) = .9804 \times (50 - 1) / 50 = .9608$$

$$S(9) = .9608 \times (49 - 1) / 49 = .9412$$

Graphing $S(t)$ against t produces a Kaplan–Meier survivor curve, like the one seen in Figure 10.1. Stata draws such graphs automatically with the **sts graph** command. For example,

```
. use C:\data\aidst, clear
. sts graph
```

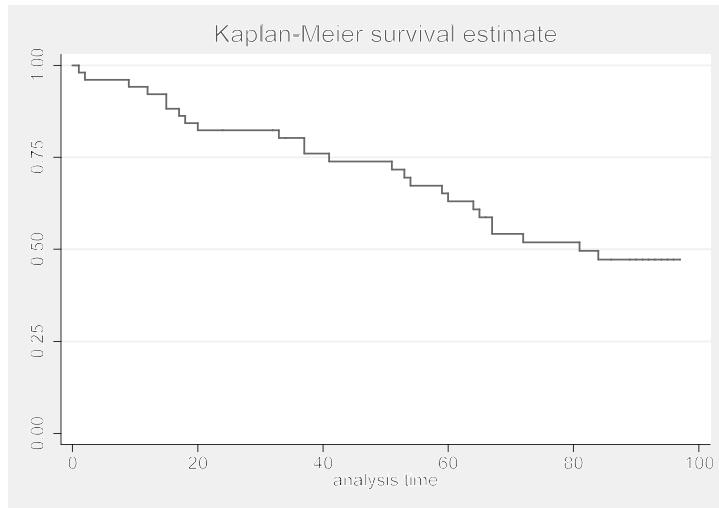


Figure 10.1

For a second example of survivor functions, we turn to data in *smoking1.dta*, adapted from Rosner (1995). The observations are data on 234 former smokers, attempting to quit. Most did not succeed. Variable *days* records how many days elapsed between quitting and starting up again. The study lasted one year, and variable *smoking* indicates whether an individual resumed smoking before the end of this study (*smoking* = 1, “failure”) or not (*smoking* = 0, “censored”). With new data, we should begin by using **stset** to set the data up for survival-time analysis.

```
. use C:\data\smoking1.dta, clear
. describe
```

Contains data from C:\data\smoking1.dta				Smoking (Rosner 1995:607)
obs:	234 <th>vars:</th> <td>8</td> <th>30 Jun 2012 10:19</th>	vars:	8	30 Jun 2012 10:19
size:	2,808			
variable	storage type	display format	value label	variable label
id	int	%9.0g		Case ID number
days	int	%9.0g		Days abstinent
smoking	byte	%9.0g		Resumed smoking
age	byte	%9.0g		Age in years
sex	byte	%9.0g	sex	Sex (female)
cigs	byte	%9.0g		Cigarettes per day
co	int	%9.0g		Carbon monoxide x 10
minutes	int	%9.0g		Minutes elapsed since last cig

Sorted by:

. stset days, failure(smoking)

```
failure event: smoking != 0 & smoking < .
obs. time interval: (0, days]
exit on or before: failure
```

234	total obs.
0	exclusions

234	obs. remaining, representing	
201	failures in single record/single failure data	
18946	total analysis time at risk, at risk from t =	0
	earliest observed entry t =	0
	last observed exit t =	366

The study involved 110 men and 124 women. Incidence rates for both sexes appear to be similar:

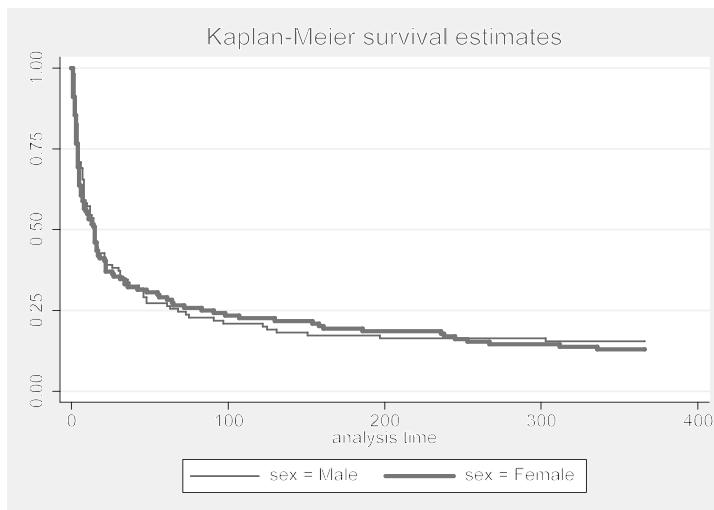
. stsum, by(sex)

```
failure _d: smoking
analysis time _t: days
```

sex	time at risk	incidence rate	no. of subjects	Survival time		
				25%	50%	75%
Male	8813	.0105526	110	4	15	68
Female	10133	.0106582	124	4	15	83
total	18946	.0106091	234	4	15	73

Figure 10.2 confirms this similarity. There appears to be little difference between the survivor functions of men and women. That is, both sexes returned to smoking at about the same rate. The survival probabilities of nonsmokers decline very steeply during the first 30 days after quitting. For either sex, there is less than a 15% chance of surviving as a nonsmoker beyond a full year.

. sts graph, by(sex) plot1opt(lwidth(medium)) plot2opt(lwidth(thick))

**Figure 10.2**

We can also formally test for the equality of survivor functions using a log-rank test. Unsurprisingly, this test finds no significant difference ($p = .6772$) between the smoking recidivism of men and women.

```
. sts test sex
failure _d: smoking
analysis time _t: days

Log-rank test for equality of survivor functions

| sex    | Events observed | Events expected |
|--------|-----------------|-----------------|
| Male   | 93              | 95.88           |
| Female | 108             | 105.12          |
| Total  | 201             | 201.00          |


chi2(1) = 0.17
Pr>chi2 = 0.6772
```

Cox Proportional Hazard Models

Regression methods allow us to take survival analysis further and examine the effects of multiple continuous or categorical predictors. One widely-used method known as Cox regression employs a proportional hazard model. The hazard rate for failure at time t is defined as the rate of failures at time t among those who have survived to time t :

$$h(t) = \frac{\text{probability of failing between times } t \text{ and } t + \Delta t}{(\Delta t) (\text{probability of failing after time } t)} \quad [10.2]$$

We model this hazard rate as a function of the baseline hazard (h_0) at time t , and the effects of one or more x variables,

$$h(t) = h_0(t) \exp(\beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k) \quad [10.3a]$$

or, equivalently,

$$\ln[h(t)] = \ln[h_0(t)] + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k \quad [10.3b]$$

“Baseline hazard” means the hazard for an observation with all x variables equal to 0. Cox regression estimates this hazard nonparametrically and obtains maximum-likelihood estimates of the β parameters in [10.3]. Stata’s **stcox** procedure ordinarily reports hazard ratios, which are estimates of $\exp(\beta)$. These indicate proportional changes relative to the baseline hazard rate.

Does age affect the onset of AIDS symptoms? Dataset *aids.dta* contains information that addresses this question. Note that with **stcox**, unlike most other Stata model-fitting commands, we list only the independent variable(s). The survival-analysis dependent variables, time variables and censoring variables are understood automatically with **stset** data.

```
. stcox age, nolog
      failure _d: aids
      analysis time _t: time
      id: case

Cox regression -- Breslow method for ties

No. of subjects =           51          Number of obs =       51
No. of failures =        25
Time at risk =           3164
Log likelihood =     -86.576295      LR chi2(1) =       5.00
                                         Prob > chi2 =    0.0254



| _t  | Haz. Ratio | Std. Err. | z    | P> z  | [95% Conf. Interval] |
|-----|------------|-----------|------|-------|----------------------|
| age | 1.084557   | .0378623  | 2.33 | 0.020 | 1.01283    1.161363  |


```

We might interpret the estimated hazard ratio, 1.084557, with reference to two HIV-positive individuals whose ages are a and $a + 1$. The older person is 8.5% more likely to develop AIDS symptoms over a short period of time (that is, the ratio of their respective hazards is 1.084557). This ratio differs significantly ($p = .02$) from 1. If we wanted to state our findings for a five-year difference in age, we could raise the hazard ratio to the fifth power:

```
. display exp(_b[age])^5
1.5005865
```

Thus, the hazard of AIDS onset is about 50% higher when the second person is five years older than the first. Alternatively, we could learn the same thing (and obtain the new confidence interval) by repeating the regression after creating a new version of *age* measured in five-year units. The **nolog noshow** options below suppress display of the iteration log and the **st**-dataset description.

```
. generate age5 = age/5
. label variable age5 "age in 5-year units"
. stcox age5, nolog noshow

Cox regression -- Breslow method for ties

No. of subjects =           51          Number of obs   =      51
No. of failures =          25          LR chi2(1)     =      5.00
Time at risk    =        3164          Prob > chi2    =  0.0254
Log likelihood  = -86.576295



| _t   | Haz. Ratio | Std. Err. | z    | P> z  | [95% Conf. Interval] |
|------|------------|-----------|------|-------|----------------------|
| age5 | 1.500587   | .2619305  | 2.33 | 0.020 | 1.065815 2.112711    |


```

Like ordinary regression, Cox models can have more than one independent variable. Dataset *heart.dta* contains survival-time data from Selvin (1995) on 35 patients with very high cholesterol levels. Variable *time* gives the number of days each patient was under observation. *coronary* indicates whether a coronary event occurred at the end of this time period (*coronary* = 1) or not (*coronary* = 0). The data also include cholesterol levels and other factors thought to affect heart disease. File *heart.dta* was previously set up for survival-time analysis by an **stset** *time, failure(coronary)* command, so we can go directly to **st** analysis.

```
. describe patient - ab

```

variable name	storage type	display format	value label	variable label
patient	byte	%9.0g		Patient ID number
time	int	%9.0g		Time in days
coronary	byte	%9.0g		Coronary event (1) or none (0)
weight	int	%9.0g		Weight in pounds
sbp	int	%9.0g		Systolic blood pressure
chol	int	%9.0g		Cholesterol level
cigs	byte	%9.0g		Cigarettes smoked per day
ab	byte	%9.0g		Type A (1) or B (0) personality

```
. stdescribe
failure _d: coronary
analysis time _t: time
```

Category	total	per subject			
		mean	min	median	max
no. of subjects	35				
no. of records	35	1	1	1	1
(first) entry time		0	0	0	0
(final) exit time	2580.629	773	2875	3141	
subjects with gap	0				
time on gap if gap	0				
time at risk	90322	2580.629	773	2875	3141
failures	8	.2285714	0	0	1

Cox regression finds that cholesterol level and cigarettes both significantly increase the hazard of a coronary event. Counterintuitively, weight appears to decrease the hazard. Systolic blood pressure and A/B personality do not have significant net effects.

```
. stcox weight sbp chol cigs ab, noshow nolog

Cox regression -- no ties

No. of subjects =           35                               Number of obs   =      35
No. of failures =          8                                LR chi2(5)     =    13.97
Time at risk    =  90322                                         Prob > chi2    =  0.0158
Log likelihood  = -17.263231
```

<u>t</u>	Haz. Ratio	Std. Err.	z	P> z	[95% Conf. Interval]
weight	.9349336	.0305184	-2.06	0.039	.8769919 .9967034
sbp	1.012947	.0338061	0.39	0.700	.9488087 1.081421
chol	1.032142	.0139984	2.33	0.020	1.005067 1.059947
cigs	1.203335	.1071031	2.08	0.038	1.010707 1.432676
ab	3.04969	2.985616	1.14	0.255	.4476492 20.77655

After estimating the model, we can **predict** new variables holding the estimated baseline cumulative hazard and survivor functions. Since “baseline” refers to a situation with all x variables equal to zero, however, we first need to center some variables so that 0 values make sense. A patient who weighs 0 pounds, or has 0 blood pressure, does not provide a useful comparison. Guided by the minimum values actually in our data, we might shift *weight* so that 0 indicates 120 pounds, *sbp* so that 0 indicates 105, and *chol* so that 0 indicates 340:

```
. summarize patient - ab
```

Variable	Obs	Mean	Std. Dev.	Min	Max
patient	35	18	10.24695	1	35
time	35	2580.629	616.0796	773	3141
coronary	35	.2285714	.426043	0	1
weight	35	170.0857	23.55516	120	225
sbp	35	129.7143	14.28403	104	154
chol	35	369.2857	51.32284	343	645
cigs	35	17.14286	13.07702	0	40
ab	35	.5142857	.5070926	0	1

```
. replace weight = weight - 120
. replace sbp = sbp - 105
. replace chol = chol - 340
. summarize patient - ab
```

Variable	Obs	Mean	Std. Dev.	Min	Max
patient	35	18	10.24695	1	35
time	35	2580.629	616.0796	773	3141
coronary	35	.2285714	.426043	0	1
weight	35	50.08571	23.55516	0	105
sbp	35	24.71429	14.28403	-1	49
chol	35	29.28571	51.32284	3	305
cigs	35	17.14286	13.07702	0	40
ab	35	.5142857	.5070926	0	1

Zero values for all the x variables now make real-world sense. To create new variables holding the baseline survivor and cumulative hazard function estimates, we repeat the regression and follow this by two **predict** commands:

```

. stcox weight sbp chol cigs ab, noshow nolog
. predict hazard, basehazard
. predict survivor, basesurv

Cox regression -- no ties

No. of subjects =           35          Number of obs =        35
No. of failures =          8           Time at risk =    90322
Log likelihood = -17.263231          LR chi2(5) =      13.97
                                         Prob > chi2 = 0.0158



| _t     | Haz. Ratio | Std. Err. | z     | P> z  | [95% Conf. Interval] |
|--------|------------|-----------|-------|-------|----------------------|
| weight | .9349336   | .0305184  | -2.06 | 0.039 | .8769919 .9967034    |
| sbp    | 1.012947   | .0338061  | 0.39  | 0.700 | .9488087 1.081421    |
| chol   | 1.032142   | .0139984  | 2.33  | 0.020 | 1.005067 1.059947    |
| cigs   | 1.203335   | .1071031  | 2.08  | 0.038 | 1.010707 1.432676    |
| ab     | 3.04969    | 2.985616  | 1.14  | 0.255 | .4476492 20.77655    |


```

Note that centering three x variables had no effect on the hazard ratios, standard errors and so forth. The **predict** commands created two new variables, arbitrarily named *hazard* and *survivor*. To graph the baseline survivor function, we plot *survivor* against *time* and connect data points in a staircase fashion, as seen in Figure 10.3.

```
. graph twoway line survivor time, connect(stairstep) sort
```

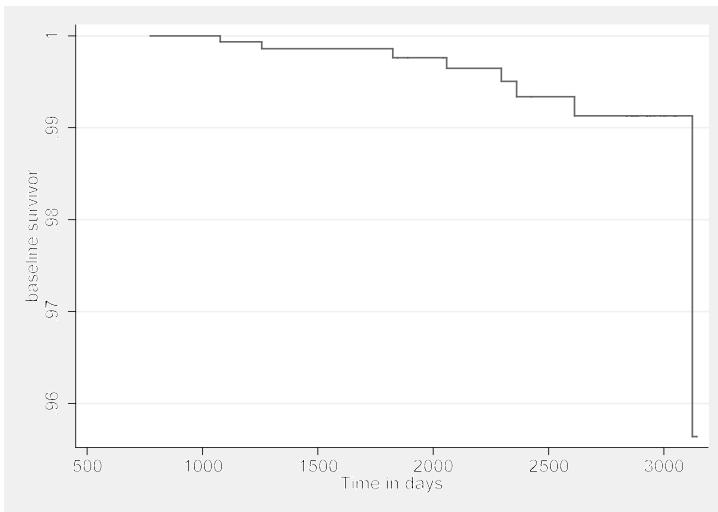


Figure 10.3

The baseline survivor function — which depicts survival probabilities for patients having “0” weight (120 pounds), “0” blood pressure (105), “0” cholesterol (340), 0 cigarettes per day, and a type B personality — declines with time. Although this decline looks precipitous at the right, notice that the probability really only falls from 1 to about .96. Given less favorable values of the predictor variables, the survival probabilities would fall much faster.

The same baseline survivor-function graph could have been obtained another way, without **stcox**. The alternative, shown in Figure 10.4, employs an **sts graph** command with **adjustfor()** option listing the predictor variables.

```
. sts graph, adjustfor(weight sbp chol cigs ab)
```

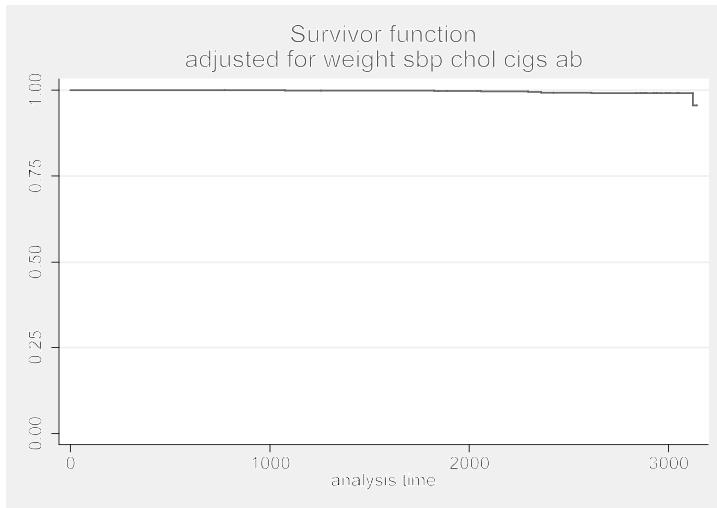


Figure 10.4

Figure 10.4, unlike Figure 10.3, follows the usual survivor-function convention of scaling the vertical axis from 0 to 1. Apart from this difference in scaling, Figures 10.3 and 10.4 depict the same curve.

Figure 10.5 graphs the estimated baseline cumulative hazard against time, using the variable (*hazard*) generated by our **stcox** command. This graph shows the baseline cumulative hazard increasing in 8 steps (because 8 patients “failed” or had coronary events), from near 0 to .033.

```
. graph twoway connected hazard time, connect(stairstep) sort
msymbol(Oh)
```

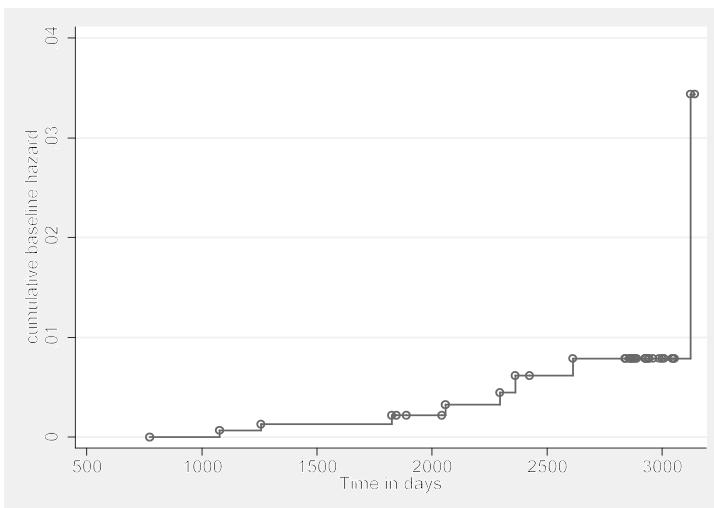


Figure 10.5

Exponential and Weibull Regression

Cox regression estimates the baseline survivor function empirically without reference to any theoretical distribution. Several alternative parametric approaches begin instead from assumptions that survival times do follow a known theoretical distribution. Possible distribution families include the exponential, Weibull, lognormal, log-logistic, Gompertz or generalized gamma. Models based on any of these can be fit through the **streg** command. Such models have the same general form as Cox regression (equations [10.2] and [10.3]), but define the baseline hazard $h_0(t)$ differently. Two examples appear in this section.

If failures occur independently, with a constant hazard, then survival times follow an exponential distribution and could be analyzed by *exponential regression*. Constant hazard means that the individuals studied do not “age,” in the sense that they are no more or less likely to fail late in the period of observation than they were at its start. Over the long term, this assumption seems unjustified for machines or living organisms, but it might approximately hold if the period of observation covers a relatively small fraction of their life spans. An exponential model implies that logarithms of the survivor function, $\ln(S(t))$, are linearly related to t .

A second common parametric approach, *Weibull regression*, is based on the more general Weibull distribution. This does not require failure rates to remain constant, but allows them to increase or decrease smoothly over time. The Weibull model implies that $\ln(-\ln(S(t)))$ is a linear function of $\ln(t)$.

Graphs provide a useful diagnostic for the appropriateness of exponential or Weibull models. For example, returning to *aids.dta*, we construct a graph (Figure 10.6) of $\ln(S(t))$ versus time, after first generating Kaplan–Meier estimates of the survivor function $S(t)$. The *y*-axis labels in Figure 10.6 are given a fixed two-digit, one-decimal display format (%2.1f) and oriented horizontally, to improve their readability.

```
. sts gen S = S
. generate logS = ln(S)
. graph twoway scatter logS time,
    ylabel(-.8(.1)0, format(%2.1f) angle(horizontal))
```

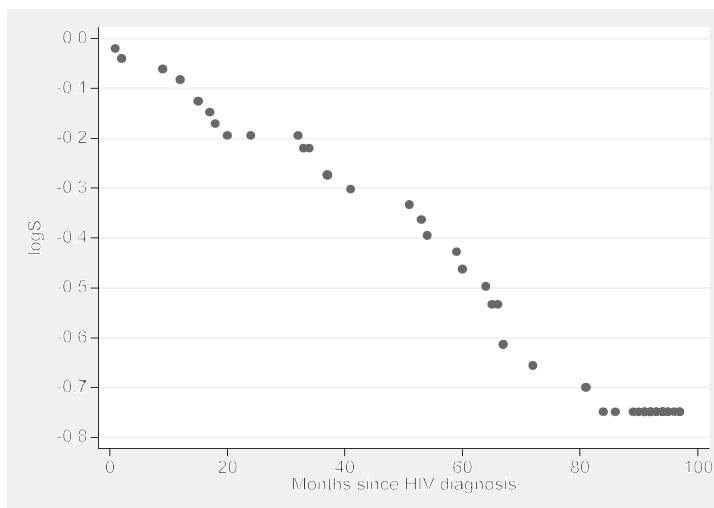


Figure 10.6

The pattern in Figure 10.6 appears somewhat linear, encouraging us to try an exponential regression:

```
. streg age, dist(exponential) nolog noshow
Exponential regression -- log relative-hazard form

No. of subjects =           51          Number of obs =      51
No. of failures =          25          LR chi2(1)      =     4.34
Time at risk =            3164          Prob > chi2    =    0.0372
Log Likelihood = -59.996976


```

<i>_t</i>	Haz. Ratio	Std. Err.	<i>z</i>	<i>P> z </i>	[95% Conf. Interval]
age	1.074414	.0349626	2.21	0.027	1.008028 1.145172
_cons	.0006811	.0007954	-6.24	0.000	.000069 .0067191

The hazard ratio (1.074) and standard error (.035) estimated by this exponential regression do not greatly differ from their counterparts (1.085 and .038) in our earlier Cox regression. The similarity reflects the degree of correspondence between empirical hazard function and the

constant hazard implied by an exponential distribution. According to this exponential model, the hazard of an HIV-positive individual developing AIDS increases about 7.4% with each year of age.

After **streg**, the **stcurve** command draws a graph of the models' cumulative hazard, survival or hazard functions. By default, **stcurve** draws these curves holding all x variables in the model at their means. We can specify other x values by using the **at()** option. The individuals in *aids.dta* ranged from 26 to 50 years old. We could graph the survival function at $age = 26$ by issuing a command such as

```
. stcurve, surviv at(age = 26)
```

A more interesting graph uses the **at1()** and **at2()** options to show the survival curve at two different sets of x values, such as the low and high extremes of age :

```
. stcurve, survival at1(age = 26) at2(age = 50) lpattern(dash solid)
```

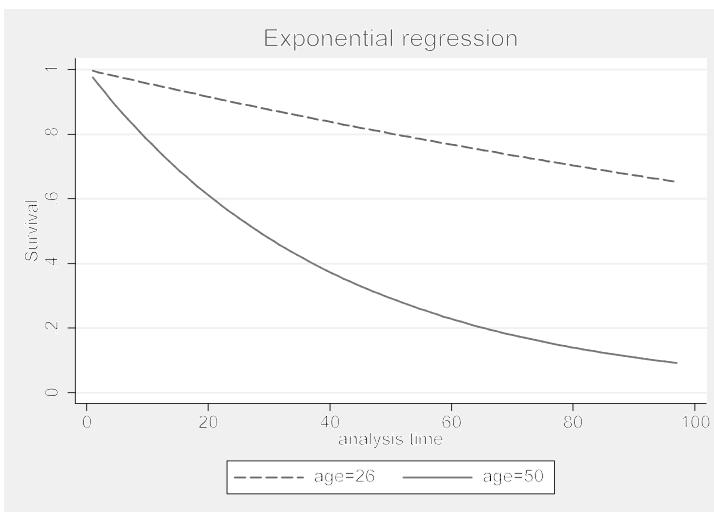


Figure 10.7

Figure 10.7 shows the predicted survival curve (for transition from HIV diagnosis to AIDS) falling more steeply among older patients. The significant age hazard ratio greater than 1 in our exponential regression table implied the same thing, but using **stcurve** with **at1()** and **at2()** values gives a strong visual interpretation of this effect. These options work in a similar manner with all three types of **stcurve** graphs:

stcurve, survival	Survival function.
stcurve, hazard	Hazard function.
stcurve, cumhaz	Cumulative hazard function.

Instead of the exponential distribution, **streg** can also fit survival models based on the Weibull distribution. A Weibull distribution might appear curvilinear in a plot of $\ln(S(t))$ versus t , but

it should be linear in a plot of $\ln(-\ln(S(t)))$ versus $\ln(t)$, such as Figure 10.8. An exponential distribution, on the other hand, will appear linear in both plots and have a slope equal to 1 in the $\ln(-\ln(S(t)))$ versus $\ln(t)$ plot. In fact, the data points in Figure 10.8 are not far from a line with slope 1, suggesting that our previous exponential model is adequate.

```
. generate loglogS = ln(-ln(S))
. generate logtime = ln(time)
. graph twoway scatter loglogS logtime, ylabel(,angle(horizontal))
```

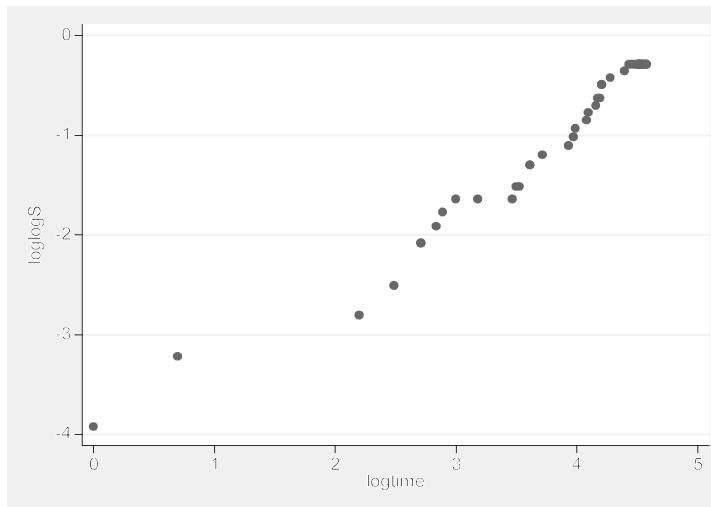


Figure 10.8

Although we do not need the additional complexity of a Weibull model with these data, results are given below for illustration.

```
. streg age, dist(weibull) noshow nolog
Weibull regression -- log relative-hazard form

No. of subjects =           51                               Number of obs   =      51
No. of failures =          25                               LR chi2(1)     =     4.68
Time at risk    =       3164                             Prob > chi2    =  0.0306
Log Likelihood = -59.778257
```

_t	Haz. Ratio	Std. Err.	z	P> z	[95% Conf. Interval]
age	1.079477	.0363509	2.27	0.023	1.010531 1.153127
_cons	.0003313	.0005415	-4.90	0.000	.00000135 .0081564
/ln_p	.1232638	.1820858	0.68	0.498	-.2336179 .4801454
p	1.131183	.2059723			.7916643 1.616309
1/p	.8840305	.1609694			.6186934 1.263162

The Weibull regression obtains a hazard ratio estimate (1.079) intermediate between our previous Cox and exponential results. The most noticeable difference from those earlier models

is the presence of three new lines at the bottom of the table. These refer to the Weibull distribution shape parameter p . A p value of 1 corresponds to an exponential model: the hazard does not change with time. $p > 1$ indicates that the hazard increases with time; $p < 1$ indicates that the hazard decreases. A 95% confidence interval for p ranges from .79 to 1.62, so we have no reason to reject an exponential ($p = 1$) model here. Different, but mathematically equivalent, parameterizations of the Weibull model focus on $\ln(p)$, p or $1/p$, so Stata provides all three. **stcurve** draws survival, hazard, or cumulative hazard functions after **streg**, **dist(weibull)** just as it does after **streg**, **dist(exponential)** or other **streg** models.

Exponential or Weibull regression is preferable to Cox regression when survival times actually follow an exponential or Weibull distribution. When they do not, these models are misspecified and can yield misleading results. Cox regression, which makes no *a priori* assumptions about distribution shape, remains useful in a wider variety of situations.

In addition to exponential and Weibull models, **streg** can fit models based on the Gompertz, lognormal, log-logistic or generalized gamma distributions. Type **help streg** or consult the *Survival Analysis and Epidemiological Tables Reference Manual* for syntax and a list of options.

Poisson Regression

If events occur independently and with constant rate, then counts of events over a given period of time follow a Poisson distribution. Let r_j represent the incidence rate:

$$r_j = \frac{\text{count of events}}{\text{number of times event could have occurred}} \quad [10.4]$$

The denominator in [10.4] is termed the “exposure” and is often measured in units such as person-years. We model the logarithm of incidence rate as a linear function of one or more predictors:

$$\ln(r_j) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k \quad [10.5a]$$

Equivalently, the model describes logs of expected event counts:

$$\ln(\text{expected count}) = \ln(\text{exposure}) + \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k \quad [10.5b]$$

Assuming that a Poisson process underlies the events of interest, Poisson regression finds maximum-likelihood estimates of the β parameters.

Data on radiation exposure and cancer deaths among workers at Oak Ridge National Laboratory provide an example. The 56 observations in dataset *oakridge.dta* represent 56 age/radiation-exposure categories (7 categories of age \times 8 categories of radiation exposure). For each combination, we know the number of deaths and the number of person-years of exposure.

```
. use C:\data\oakridge.dta, clear
```

```
. describe
```

Contains data from C:\data\oakridge.dta
 obs: 56
 vars: 4
 size: 392

Radiation (Selvin 1995:474)
 30 Jun 2012 10:19

variable	storage type	display format	value label	variable label
age	byte	%9.0g	ageg	Age group
rad	byte	%9.0g		Radiation exposure level
deaths	byte	%9.0g		Number of deaths
pyears	float	%9.0g		Person-years

Sorted by:

```
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
age	56	4	2.0181	1	7
rad	56	4.5	2.312024	1	8
deaths	56	1.839286	3.178203	0	16
pyears	56	3807.679	10455.91	23	71382

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

	age	rad	deaths	pyears
1.	< 45	1	0	29901
2.	45-49	1	1	6251
3.	50-54	1	4	5251
4.	55-59	1	3	4126
5.	60-64	1	3	2778
6.	65-69	1	1	1607

Does the death rate increase with exposure to radiation? Poisson regression finds a statistically significant effect:

```
. poisson deaths rad, nolog exposure(pyyears) irr
```

Poisson regression
 Number of obs = 56
 LR chi2(1) = 14.87
 Prob > chi2 = 0.0001
 Pseudo R2 = 0.0420
 Log likelihood = -169.7364

deaths	IRR	Std. Err.	z	P> z	[95% Conf. Interval]
rad	1.236469	.0603551	4.35	0.000	1.123657 1.360606
_cons	.000288	.0000483	-48.65	0.000	.0002074 .0004
ln(pyyears)	1	(exposure)			

For the regression above, we specified the event count (*deaths*) as the dependent variable and radiation (*rad*) as the independent variable. The Poisson exposure variable is *pyyears*, or person-years in each category of *rad*. The *irr* option calls for incidence rate ratios rather than regression coefficients in the results table — that is, we get estimates of $\exp(\beta)$ instead of β , the default. According to this incidence rate ratio, the death rate becomes 1.236 times higher (increases by 23.6%) with each increase in radiation category. Although that ratio is statistically significant, the fit is not impressive. The pseudo R^2 (equation [9.4]) is only .042.

To perform a goodness-of-fit test, comparing the Poisson model's predictions with the observed counts, use the postestimation command **estat gof**:

```
. estat gof
Deviance goodness-of-fit = 254.5475
Prob > chi2(54)          = 0.0000
Pearson goodness-of-fit  = 419.0209
Prob > chi2(54)          = 0.0000
```

These goodness-of-fit test results indicate that our model's predictions are significantly different from the actual counts — another sign that the model fits poorly.

We obtain better results when we include *age* as a second predictor. Pseudo R^2 then rises to .5966, and the goodness-of-fit test no longer leads to rejecting our model.

```
. poisson deaths rad age, nolog exposure(pyears) irr
Poisson regression
Number of obs      =      56
LR chi2(2)         =    211.41
Prob > chi2        =  0.0000
Pseudo R2          =  0.5966
Log likelihood = -71.4653
```

deaths	IRR	Std. Err.	z	P> z	[95% Conf. Interval]
rad	1.176673	.0593446	3.23	0.001	1.065924 1.298929
age	1.960034	.0997536	13.22	0.000	1.773955 2.165631
_cons	.0000297	9.05e-06	-34.18	0.000	.0000163 .0000539
ln(pyears)	1	(exposure)			

```
. poisgof
Deviance goodness-of-fit = 58.00534
Prob > chi2(53)          = 0.2960
Pearson goodness-of-fit  = 51.91816
Prob > chi2(53)          = 0.5163
```

For simplicity, to this point we have treated *rad* and *age* as if both were continuous variables, and we expect their effects on the log death rate to be linear. In fact, however, both independent variables are measured as ordered categories. *rad* = 1, for example, means 0 radiation exposure; *rad* = 2 means 0 to 19 millisieverts; *rad* = 3 means 20 to 39 millisieverts; and so forth. An alternative way to include radiation exposure categories in the regression, while watching for nonlinear effects, is as a set of indicator variables using Stata's factor-variable notation. The *i.rad* term in the following model specifies that a {0, 1} indicator for each category of *rad* be included along with *age* as predictors. *rad* = 1 is automatically omitted as the base category.

. poisson deaths i.rad age, nolog exposure(pyears) irr						
Poisson regression			Number of obs = 56			
			LR chi2(8) = 215.44			
			Prob > chi2 = 0.0000			
Log likelihood = -69.451814			Pseudo R2 = 0.6080			
deaths	IRR	Std. Err.	z	P> z 	[95% Conf. Interval]	
rad						
2	1.473591	.426898	1.34	0.181	.8351884	2.599975
3	1.630688	.6659257	1.20	0.231	.732428	3.630587
4	2.375967	1.088835	1.89	0.059	.9677429	5.833389
5	.7278113	.7518255	-0.31	0.758	.0961018	5.511957
6	1.168477	1.20691	0.15	0.880	.1543195	8.847472
7	4.433729	3.337738	1.98	0.048	1.013863	19.38915
8	3.89188	1.640978	3.22	0.001	1.703168	8.893267
age	1.961907	.1000652	13.21	0.000	1.775267	2.168169
_cons	.0000295	.0000106	-29.03	0.000	.0000146	.0000597
ln(pyears)	1	(exposure)				

The additional complexity of this indicator-variable model brings little improvement in fit. It does, however, add to our interpretation. The overall effect of radiation on death rate appears to come primarily from the two highest radiation levels ($rad = 7$ and $rad = 8$, corresponding to 100 to 119 and 120 or more milliseverts). At these levels, the incidence rates are about four times higher.

Radiation levels 7 and 8 seem to have similar effects, so we might simplify the model by combining them. First, we test whether their coefficients are significantly different. They are not:

```
. test 7.rad = 8.rad
( 1) [deaths]7.rad - [deaths]8.rad = 0
      chi2( 1) =    0.03
      Prob > chi2 =  0.8676
```

Next, generate a new dummy variable $rad78$, which equals 1 if rad equals 7 or 8. Substitute this new variable for the separate $rad = 7$ and $rad = 8$ indicators. The commands illustrate how to do so in factor-variable notation.

```
. generate rad78 = (7.rad | 8.rad)
. poisson deaths i(1/6).rad rad78 age, irr ex(pyears) nolog
```

Poisson regression		Number of obs = 56				
		LR chi2(7) = 215.41				
		Prob > chi2 = 0.0000				
		Pseudo R2 = 0.6079				
Log likelihood = -69.465332						
deaths		IRR	Std. Err.	z	P> z	[95% Conf. Interval]
rad						
2	1.473602	.4269013	1.34	0.181	.8351949	2.599996
3	1.630718	.6659381	1.20	0.231	.7324415	3.630655
4	2.376065	1.08888	1.89	0.059	.9677823	5.833629
5	.7278387	.7518538	-0.31	0.758	.0961055	5.512165
6	1.168507	1.206942	0.15	0.880	.1543236	8.847704
rad78	3.980326	1.580024	3.48	0.001	1.828214	8.665833
age	1.961722	.100043	13.21	0.000	1.775122	2.167937
_cons	.0000296	.0000106	-29.03	0.000	.0000146	.0000598
ln(pyears)	1	(exposure)				

We could proceed to simplify the model further in this fashion. At each step, **test** helps to evaluate whether combining two dummy variables is justifiable.

Generalized Linear Models

Generalized linear models (GLM) have the form

$$g[\text{E}(y)] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k, \quad y \sim F \quad [10.6]$$

where $g[\]$ is the *link function* and F the distribution family. This general formulation encompasses many specific models. For example, if $g[\]$ is the identity function and y follows a normal (Gaussian) distribution, we have a linear regression model:

$$\text{E}(y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k, \quad y \sim \text{Normal} \quad [10.7]$$

If $g[\]$ is the logit function and y follows a Bernoulli distribution, we have logit regression instead:

$$\text{logit}[\text{E}(y)] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k, \quad y \sim \text{Bernoulli} \quad [10.8]$$

Because of its broad applications, GLM could have been introduced at several different points in this book. Its relevance to this chapter comes from the ability to fit event models. Poisson regression, for example, requires that $g[\]$ is the natural log function and that y follows a Poisson distribution:

$$\ln[\text{E}(y)] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k, \quad y \sim \text{Poisson} \quad [10.9]$$

As might be expected with such a flexible method, Stata's **glm** command permits many different options. Users can specify not only the distribution family and link function, but also details of the variance estimation, fitting procedure, output and exposure. These options make **glm** a

useful alternative even when applied to models for which a dedicated command (such as **regress**, **logistic** or **poisson**) already exists.

We might represent a generic **glm** command as follows:

```
. glm y x1 x2 x3, family(familyname) link(linkname)
    exposure(expvar) eform vce(jackknife)
```

where **family()** specifies the *y* distribution family, **link()** the link function, and **exposure()** an exposure variable such as that needed for Poisson regression. The **eform** option asks for regression coefficients in exponentiated form, $\exp(\beta)$ rather than β . Standard errors are estimated through jackknife calculations.

Possible distribution families are

family(gaussian)	Gaussian or normal (default)
family(igaussian)	Inverse Gaussian
family(binomial)	Bernoulli binomial
family(poisson)	Poisson
family(nbinomial)	Negative binomial
family(gamma)	Gamma

We can also specify a number or variable indicating the binomial denominator *N* (number of trials), or a number indicating the negative binomial variance and deviance functions, by declaring them in the **family()** option:

```
family(binomial #)
family(binomial varname)
family(nbinomial #)
```

Possible link functions are

link(identity)	Identity (default)
link(log)	Log
link(logit)	Logit
link(probit)	Probit
link(cloglog)	Complementary log-log
link(opower #)	Odds power
link(power #)	Power
link(nbinomial)	Negative binomial
link(loglog)	Log-log
link(logc)	Log-complement

Coefficient variances or standard errors can be estimated in a variety of ways. A partial list of **glm** variance-estimating options is given below:

opg	Berndt, Hall, Hall and Hausman B-H-cubed variance estimator.
oim	Observed information matrix variance estimator.
robust	Huber/White/sandwich estimator of variance.
unbiased	Unbiased sandwich estimator of variance
nwest	Heteroskedasticity and autocorrelation-consistent variance estimator.

jackknife Jackknife estimate of variance.

jackknife1 One-step jackknife estimate of variance.

bootstrap Bootstrap estimate of variance. The default is 199 repetitions; specify some other number by adding the **bsrep(#)** option.

For a full list of options with some technical details, look up **glm** in the *Base Reference Manual*. A more in-depth treatment of GLM topics can be found in Hardin and Hilbe (2012).

Chapter 7 began with the simple regression of life expectancy on the mean years of schooling in 188 nations:

```
. use C:\data\Nations2.dta, clear
. regress life school
```

Source	SS	df	MS	Number of obs	=	188
Model	9846.65406	1	9846.65406	F(1, 186)	=	206.34
Residual	8875.86926	186	47.7197272	Prob > F	=	0.0000
Total	18722.5233	187	100.120446	R-squared	=	0.5259
				Adj R-squared	=	0.5234
				Root MSE	=	6.9079

life	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
school	2.45184	.1706856	14.36	0.000	2.115112 2.788569
_cons	50.35941	1.36924	36.78	0.000	47.65817 53.06065

We can fit the same model and obtain the same estimates via a **glm** command,

```
. glm life school, link(identity) family(gaussian)

Iteration 0:  log likelihood = -629.09751

Generalized linear models
Optimization : ML
No. of obs      =      188
Residual df    =      186
Scale parameter = 47.71973
Deviance       = 8875.869256
(1/df) Deviance = 47.71973
Pearson        = 8875.869256
(1/df) Pearson = 47.71973

Variance function: V(u) = 1
[Gaussian]
Link function   : g(u) = u
[Identity]

AIC            = 6.713803
BIC            = 7901.891

Log likelihood  = -629.0975058
```


life	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
school	2.45184	.1706856	14.36	0.000	2.117303 2.786378
_cons	50.35941	1.36924	36.78	0.000	47.67575 53.04307

Because **link(identity)** and **family(gaussian)** are default options, we could have left them out of the previous **glm** command.

We could also fit the same OLS model but obtain bootstrap standard errors.

```
. glm life school, link(identity) family(gaussian) vce(bootstrap)
(running glm on estimation sample)

Bootstrap replications (50)
----- 1 2 3 4 5
..... 50

Generalized linear models          No. of obs     =      188
Optimization : ML                 Residual df    =      186
Deviance      =  8875.869256      Scale parameter =  47.71973
Pearson       =  8875.869256      (1/df) Deviance =  47.71973
                                         (1/df) Pearson  =  47.71973

Variance function: V(u) = 1        [Gaussian]
Link function : g(u) = u          [Identity]

Log Likelihood = -629.0975058    AIC            =  6.713803
                                         BIC            =  7901.891


```

	Observed Coef.	Bootstrap Std. Err.	z	P> z	Normal-based [95% Conf. Interval]	
life	2.45184	.1385655	17.69	0.000	2.180257	2.723424
school _cons	50.35941	1.230463	40.93	0.000	47.94775	52.77108

The bootstrap standard errors reflect observed variation among coefficients estimated from 50 samples of $n = 188$ cases each, drawn by random sampling with replacement from the original $n = 188$ dataset. In this example, the bootstrap standard errors are less than the corresponding theoretical standard errors, and the resulting confidence intervals are narrower.

Similarly, we could use **glm** to repeat the logistic regression with the space shuttle data that began Chapter 9. For this example we ask for jackknife standard errors and odds ratio or exponential-form (**eform**) coefficients:

```
. use C:\data\shuttle0.dta, clear
. glm any date, link(logit) family(bernoulli) eform vce(jackknife)
(running glm on estimation sample)

Jackknife replications (23)
----- 1 2 3 4 5
..... 23

Generalized linear models          No. of obs     =      23
Optimization : ML                 Residual df    =      21
Deviance      =  25.98219269      Scale parameter =  1
Pearson       =  22.8885488      (1/df) Deviance =  1.237247
                                         (1/df) Pearson  =  1.089931

Variance function: V(u) = u*(1-u/1)      [Binomial]
Link function : g(u) = ln(u/(1-u))      [Logit]

Log Likelihood = -12.99109634    AIC            =  1.303574
                                         BIC            = -39.86319


```

	Odds Ratio	Jackknife Std. Err.	t	P> t	[95% Conf. Interval]	
any	1.002093	.0015797	1.33	0.198	.9988222	1.005374
date _cons	1.34e-08	1.89e-07	-1.28	0.214	2.32e-21	76840.52

The **poisson** regression with indicator variables presented earlier in this chapter,

```
. use C:\data\oakridge.dta, clear
. poisson deaths i.rad age, nolog exposure(pyears) irr
```

corresponds to the following **glm** model:

```
. glm deaths i.rad age, link(log) family(poisson) exposure(pyears)
eform

Iteration 0:  log likelihood = -75.68551
Iteration 1:  log likelihood = -69.595462
Iteration 2:  log likelihood = -69.452909
Iteration 3:  log likelihood = -69.451814
Iteration 4:  log likelihood = -69.451814

Generalized linear models                                No. of obs      =      56
Optimization     : ML                                    Residual df     =      47
                                                               Scale parameter =      1
Deviance        =  53.97836926                         (1/df) Deviance =  1.148476
Pearson         =  53.59824023                         (1/df) Pearson  =  1.140388

Variance function: v(u) = u                           [Poisson]
Link function   : g(u) = ln(u)                         [Log]

Log Likelihood  = -69.451814                         AIC            =  2.801851
                                                BIC            = -135.2132
```

deaths	OIM					
	IRR	Std. Err.	z	P> z	[95% Conf. Interval]	
rad						
2	1.473591	.426898	1.34	0.181	.8351884	2.599975
3	1.630688	.6659257	1.20	0.231	.732428	3.630587
4	2.375967	1.088835	1.89	0.059	.9677429	5.833389
5	.7278114	.7518256	-0.31	0.758	.0961019	5.511958
6	1.168477	1.20691	0.15	0.880	.1543196	8.847472
7	4.433727	3.337737	1.98	0.048	1.013862	19.38915
8	3.89188	1.640978	3.22	0.001	1.703168	8.893267
age	1.961907	.1000652	13.21	0.000	1.775267	2.168169
_cons	.0000295	.0000106	-29.03	0.000	.0000146	.0000597
ln(pyears)	1	(exposure)				

Although **glm** can replicate the models fit by many specialized commands, and adds some new capabilities, the specialized commands have their own advantages including speed and customized options. A particular attraction of **glm** is its ability to fit models for which Stata has no specialized command.

Principal Component, Factor and Cluster Analysis

Principal component and factor analysis provide methods for simplification, combining many correlated variables into a smaller number of underlying dimensions. Along the way to achieving simplification, the analyst must choose from a daunting variety of options. If the data really do reflect distinct underlying dimensions, different options might nonetheless converge on similar results. In the absence of distinct underlying dimensions, however, different options can lead to divergent results. Experimenting with these options suggests how stable a particular finding is, or how much it depends on arbitrary choices about the analytical technique.

Stata accomplishes principal component and factor analysis with five basic commands:

- pca** Principal component analysis.
- factor** Extracts factors of several different types, including principal components.
- screeplot** Constructs a scree graph (plot of the eigenvalues) from the recent **pca** or **factor**.
- rotate** Performs orthogonal (uncorrelated factors) or oblique (correlated factors) rotation, after **factor**.
- predict** Generates factor scores (composite variables) and other case statistics after **pca**, **factor** or **rotate**.

The factor scores or composite variables generated by **predict** can subsequently be saved, listed, graphed or analyzed like any other Stata variable. A new section illustrates such analysis with a survey-data example.

Users who create composite variables by the rough method of adding other variables together without doing factor analysis could assess their results by calculating an α (alpha) reliability coefficient:

- alpha** Cronbach's α reliability

Instead of combining variables, cluster analysis combines observations by finding non-overlapping, empirically-based typologies or groups. Cluster analysis methods are even more diverse than those of factor analysis. Stata's **cluster** command provides tools for performing cluster analysis, graphing the results, and forming new variables to identify the resulting groups. Principal component, factor analysis, cluster analysis and related commands are detailed in Stata's *Multivariate Statistics* reference manual.

The chapter concludes with a second look at Stata's structural equation modeling (**sem**) capabilities, applied to measurement models or models that include measurement-model components.

Methods described in this chapter can be accessed through the following menus:

Statistics > Multivariate analysis

Graphics > Multivariate analysis graphs

Statistics > SEM (structural equation modeling)

Example Commands

- **pca x1-x20**

Obtains principal components of the variables *x1* through *x20*.

- **pca x1-x20, mineigen(1)**

Obtains principal components of the variables *x1* through *x20*. Retains components having eigenvalues greater than 1.

- **factor x1-x20, ml factor(5)**

Performs maximum likelihood factor analysis of the variables *x1* through *x20*. Retains only the first five factors.

- **screeplot**

Scree plot or graph of eigenvalues versus factor or component number, from the most recent **factor** command.

- **rotate, varimax factors(2)**

Performs orthogonal (varimax) rotation of the first two factors from the most recent **factor** command.

- **rotate, promax factors(3)**

Performs oblique (promax) rotation of the first three factors from the most recent **factor** command.

- **predict f1 f2 f3**

Generates three new factor score variables named *f1*, *f2* and *f3*, based upon the most recent **factor** and **rotate** commands.

- **alpha x1-x10**

Calculates Cronbach's α reliability coefficient for a composite variable defined as the sum of *x1* through *x10*. The sense of items entering negatively is ordinarily reversed. Options can override this default, or form a composite variable by adding together either the original variables or their standardized values.

. cluster centroidlinkage x y z w, measure(L2) name(L2cent)

Performs agglomerative cluster analysis with centroid linkage, using variables x, y, z , and w . Euclidean distance (**L2**) measures dissimilarity among observations. Results from this cluster analysis are saved with the name *L2cent*.

**. cluster dendrogram, ylabel(0(.5)3) cutnumber(20)
 xlabel(), angle(vertical))**

Draws a cluster analysis tree graph or dendrogram showing results from the previous cluster analysis. **cutnumber(20)** specifies that the graph begins with only 20 clusters remaining, after some previous fusion of the most-similar observations. Labels are printed in a compact vertical fashion below the graph.

. cluster generate ctype = groups(3), name(L2cent)

Creates a new variable *ctype* (values of 1, 2 or 3) that classifies each observation into one of the top three groups found by the cluster analysis named *L2cent*.

Principal Component Analysis and Principal Component Factoring

To illustrate principal component and factor analysis, we start with the small dataset, *planets.dta*, describing the nine classical planets of this solar system (from Beatty et al. 1981). The data include several variables in both raw and natural logarithm form. Logarithms are employed here to reduce skew and linearize relationships among the variables.

```
. use C:\data\planets.dta, clear
. describe

Contains data from C:\data\planets.dta
obs: 9                                     Solar system data
vars: 12                                    12 Jun 2012 11:34
size: 405

variable   storage      display      value
          type        format     label
variable label
```

variable	storage	display	value	label
name	type	format	label	variable
planet	str7	%9s		Planet
dsun	float	%9.0g		Mean dist. sun, km*10^6
radius	float	%9.0g		Equatorial radius in km
rings	byte	%8.0g	ringlbl	Has rings?
moons	byte	%8.0g		Number of known moons
mass	float	%9.0g		Mass in kilograms
density	float	%9.0g		Mean density, g/cm^3
logsun	float	%9.0g		natural log dsun
lograd	float	%9.0g		natural log radius
logmoons	float	%9.0g		natural log (moons + 1)
logmass	float	%9.0g		natural log mass
logdense	float	%9.0g		natural log dense

Sorted by: dsun

Principal component analysis finds the linear combination that explains the maximum amount of variance among the observed variables — called the “first principal component.” It also finds another, orthogonal (uncorrelated) linear combination that explains the maximum amount of remaining variance (“second principal component”), and so on until all variance is explained. From k variables we extract k principal components, which between them explain all of the variance. Principle component analysis serves as a data reduction technique because fewer than

k components will often explain most of the observed variance. If further work concentrates on those components, the analysis can be simplified.

Applied to six variables describing the planets, we get six principal components that explain all the variance:

```
. pca rings logdsun - logdense
```

Principal components/correlation	Number of obs = 9
	Number of comp. = 6
	Trace = 6
Rotation: (unrotated = principal)	Rho = 1.0000

Component	Eigenvalue	Difference	Proportion	Cumulative
Comp1	4.62365	3.45469	0.7706	0.7706
Comp2	1.16896	1.05664	0.1948	0.9654
Comp3	.112323	.0539515	0.0187	0.9842
Comp4	.0583717	.0217421	0.0097	0.9939
Comp5	.0366296	.0365651	0.0061	1.0000
Comp6	.00006454	.	0.0000	1.0000

Principal components (eigenvectors)

variable	Comp1	Comp2	Comp3	Comp4	Comp5	Comp6
rings	0.4554	0.0714	0.2912	0.0351	-0.8370	0.0301
logdsun	0.3121	-0.6576	0.5930	-0.1418	0.3135	-0.0156
lograd	0.4292	0.3455	-0.0390	-0.3216	0.2619	0.7231
logmoons	0.4541	0.0003	-0.1567	0.8466	0.2286	0.0156
logmass	0.3878	0.5037	0.1374	-0.2427	0.2675	-0.6682
logdense	-0.3930	0.4352	0.7201	0.3157	0.0932	0.1708

variable	Unexplained
rings	0
logdsun	0
lograd	0
logmoons	0
logmass	0
logdense	0

The **pca** output tells us that together, the first two components explain more than 96% of the cumulative variance of all six variables. Eigenvalues correspond to the standardized variance explained by each component. With six variables, the total standardized variance is 6. Of this, we see that component 1 explains 4.62365, which amounts to $4.62365/6 = .7706$ or about 77% of the total. Component 2 explains $1.16896/6 = .1948$, or an additional 19%. Principal components having eigenvalues below 1.0 are explaining less than the equivalent of one variable's variance, which makes them unhelpful for data reduction. Analysts commonly set aside minor components and focus on those that have eigenvalues of at least one.

A good way to proceed with such data reduction is through the **factor** command, which offers principal component factoring as one of its options. To obtain principal component factors, type

. factor rings logdsun - logdense, pcf (obs=9)				
Factor analysis/correlation		Number of obs = 9		
Method: principal-component factors		Retained factors = 2		
Rotation: (unrotated)		Number of params = 11		
Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	4.62365	3.45469	0.7706	0.7706
Factor2	1.16896	1.05664	0.1948	0.9654
Factor3	0.11232	0.05395	0.0187	0.9842
Factor4	0.05837	0.02174	0.0097	0.9939
Factor5	0.03663	0.03657	0.0061	1.0000
Factor6	0.00006	.	0.0000	1.0000
LR test: independent vs. saturated: chi2(15) = 100.49 Prob>chi2 = 0.0000				
Factor loadings (pattern matrix) and unique variances				
Variable	Factor1	Factor2	Uniqueness	
rings	0.9792	0.0772	0.0353	
logdsun	0.6710	-0.7109	0.0443	
lograd	0.9229	0.3736	0.0088	
logmoons	0.9765	0.0003	0.0465	
logmass	0.8338	0.5446	0.0082	
logdense	-0.8451	0.4705	0.0644	

Principal component factoring starts with extraction of principal components, but then retains only those that meet criteria for importance — by default, those with eigenvalues above 1. As we saw in the earlier **pca** example, only the first two components here meet that criterion, and together explain over 96% of the six variables' combined variance. The unimportant 3rd through 6th principal components can safely be disregarded.

Two **factor** options provide control over the number of factors extracted:

factors(#) where # specifies the number of factors

mineigen(#) where # specifies the minimum eigenvalue for retained factors

Because principal component factoring automatically drops factors with eigenvalues below 1,

. factor rings logdsun - logdense, pcf

is equivalent to

. factor rings logdsun - logdense, pcf mineigen(1)

In this example, we would also have obtained the same results by typing

. factor rings logdsun - logdense, pcf factors(2)

To see a scree graph (plot of eigenvalues versus component or factor number) after any **factor**, use the **screeplot** command. A horizontal line at eigenvalue = 1 in Figure 11.1 marks the usual cutoff for retaining principal components, and again emphasizes the unimportance of components 3 through 6.

```
. screeplot, yline(1)
```

**Figure 11.1**

Rotation

Rotation further simplifies factor structure. After factoring, type **rotate** followed by an option to specify the type of rotation. Two common types are:

- varimax** Varimax orthogonal rotation, producing uncorrelated factors or components (default).
- promax()** Promax oblique rotation, allowing correlated factors or components. Choose a number (promax power) ≤ 4 ; the higher the number, the greater the degree of inter-factor correlation. **promax(3)** is the default.

Type **help rotate** for a complete list of rotation methods and other options. For example:

- factors()** As it does with **factor**, this option specifies how many factors to retain.
- entropy** Minimum entropy orthogonal rotation.

Rotation can be performed following any factor analysis, not just principal component factoring, as will be seen later. This section follows through with our **factor**, **pcf** example. For orthogonal (default) rotation of the first two components found in the planetary data, we simply type **rotate**.

```
. rotate
```

```
Factor analysis/correlation
  Method: principal-component factors
  Rotation: orthogonal varimax (Kaiser off)
                                         Number of obs      =         9
                                         Retained factors =          2
                                         Number of params =        11
```

Factor	Variance	Difference	Proportion	Cumulative
Factor1	3.36900	0.94539	0.5615	0.5615
Factor2	2.42361	.	0.4039	0.9654

```
LR test: independent vs. saturated: chi2(15) = 100.49 Prob>chi2 = 0.0000
```

Rotated factor loadings (pattern matrix) and unique variances

Variable	Factor1	Factor2	Uniqueness
rings	0.8279	0.5285	0.0353
logdsun	0.1071	0.9717	0.0443
lograd	0.9616	0.2580	0.0088
logmoons	0.7794	0.5882	0.0465
logmass	0.9936	0.0678	0.0082
logdense	-0.3909	-0.8848	0.0644

Factor rotation matrix

	Factor1	Factor2
Factor1	0.7980	
Factor2	0.6026	-0.7980

The example above accepts all the defaults: varimax rotation and the same number of factors retained in the last **factor**. We could have asked for the same rotation explicitly, by adding options to the command: **rotate, varimax factors(2)**.

For oblique promax rotation (allowing correlated factors) of the most recent factoring, type

```
. rotate, promax
```

```
Factor analysis/correlation
  Method: principal-component factors
  Rotation: oblique promax (Kaiser off)
                                         Number of obs      =         9
                                         Retained factors =          2
                                         Number of params =        11
```

Factor	Variance	Proportion	Rotated factors are correlated
Factor1	4.12467	0.6874	
Factor2	3.32370	0.5539	

```
LR test: independent vs. saturated: chi2(15) = 100.49 Prob>chi2 = 0.0000
```

Rotated factor loadings (pattern matrix) and unique variances

Variable	Factor1	Factor2	Uniqueness
rings	0.7626	0.3466	0.0353
logdsun	-0.1727	1.0520	0.0443
lograd	0.9926	0.0060	0.0088
logmoons	0.6907	0.4275	0.0465
logmass	1.0853	-0.2154	0.0082
logdense	-0.1692	-0.8719	0.0644

Factor rotation matrix

	Factor1	Factor2
Factor1	0.9250	0.7898
Factor2	0.3800	-0.6134

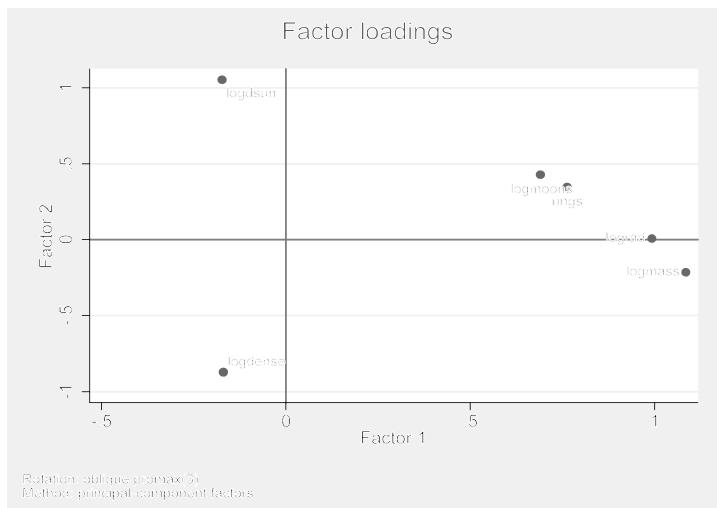
By default, this example used a promax power of 3. We could have specified the promax power and desired number of factors explicitly:

```
. rotate, promax(3) factors(2)
```

promax(4) would permit further simplification of the loading matrix, at the cost of stronger interfactor correlations and less total variance explained.

After promax rotation, *rings*, *lograd*, *logmoons* and *logmass* load most heavily on factor 1. This appears to be a “large size/many satellites” dimension. *logdsun* and *logdense* load higher on factor 2, forming a “far out/low density” dimension. A post-factor-analysis graphing command, **loadingplot**, helps to visualize these results (Figure 11.2).

```
. loadingplot, factors(2) yline(0) xline(0)
```

**Figure 11.2**

Factor Scores

Factor scores are linear composites, formed by standardizing each variable to zero mean and unit variance, and then weighting with factor score coefficients and summing for each factor. **predict** performs these calculations automatically, using the most recent **rotate** or **factor** results. In the **predict** command we supply names for the new variables, such as *f1* and *f2*.

```
. predict f1 f2
(regression scoring assumed)

Scoring coefficients (method = regression; based on promax(3) rotated
factors)
```

variable	Factor1	Factor2
rings	0.22099	0.12674
logdsun	-0.09689	0.48769
lograd	0.30608	-0.03840
logmoons	0.19543	0.16664
logmass	0.34386	-0.14338
logdense	-0.01609	-0.39127

```
. label variable f1 "Large size/many satellites"
. label variable f2 "Far out/low density". list planet f1 f2
```

	planet	f1	f2
1.	Mercury	-.9172388	-1.256881
2.	Venus	-.5160229	-1.188757
3.	Earth	-.3939372	-1.035242
4.	Mars	-.6799535	-.5970106
5.	Jupiter	1.342658	.3841085
6.	Saturn	1.184475	.9259058
7.	Uranus	.7682409	.9347457
8.	Neptune	.647119	.8161058
9.	Pluto	-1.43534	1.017025

Being standardized variables, the new factor scores *f1* and *f2* have means (approximately) equal to 0 and standard deviations equal to 1:

```
. summarize f1 f2
```

Variable	Obs	Mean	Std. Dev.	Min	Max
f1	9	-3.31e-09	1	-1.43534	1.342658
f2	9	9.93e-09	1	-1.256881	1.017025

Thus, the factor scores are measured in units of standard deviations from their means. Mercury, for example, is about .92 standard deviations below average on the large size/many satellites (*f1*) dimension because it is small and has no satellites. Mercury is 1.26 standard deviations below average on the far out/low density (*f2*) dimension because it is close to the sun and high density. Saturn, in contrast, is 1.18 and .93 standard deviations above average on these two dimensions.

Promax rotation permits correlations between factor scores:

	f1	f2
f1	1.0000	
f2	0.4974	1.0000

Scores on factor 1 have a moderate positive correlation with scores on factor 2. That is, far out/low density planets are more likely to be larger, with many satellites.

Another post-factor-analysis graphing command, **scoreplot**, draws a scatterplot of the observations' factor scores. Used with principal component factors, such graphs can help to identify multivariate outliers, or clusters of observations that stand apart from the rest. Figure 11.3 reveals three distinct types of planets.

```
. scoreplot, mlabel(planet) yline(0) xline(0)
```

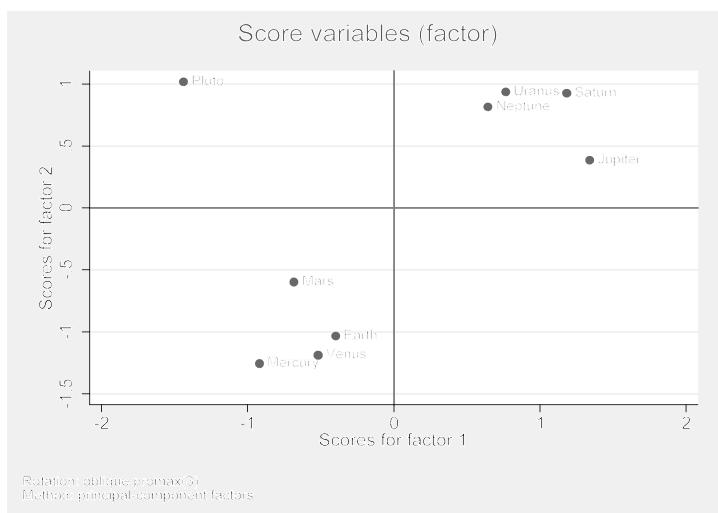


Figure 11.3

The inner, rocky planets (such as Mercury, low on “large size/many satellites” factor 1; low also on “far out/low density” factor 2) cluster together at the lower left. The outer gas giants have opposite characteristics, and cluster together at the upper right. Pluto, which physically resembles some outer-system moons, is unique among the nine classical planets for being high on the “far out/low density” dimension, and at the same time low on the “large size/many satellites” dimension. Our factor analysis thus identifies Pluto as a different kind of object that does not fit with either of the two main group of planets. Recognizing Pluto’s exceptionalism, the International Astronomical Union voted in 2006 to reclassify it from being a true planet to one of the dozens of known “dwarf planets,” leaving us with only eight true planets.

If we employ varimax instead of promax rotation, we get uncorrelated factor scores:

```
. quietly factor rings logdsun - logdense, pcf
```

```
. quietly rotate
. quietly predict varimax1 varimax2
. correlate varimax1 varimax2
(obs=9)
```

	varimax1	varimax2
varimax1	1.0000	
varimax2	0.0000	1.0000

Once created by **predict**, factor scores can be treated like any other Stata variable — listed, correlated, graphed and so forth. Factor scores often are used in social and behavioral science to combine many test or questionnaire items into composite variables or indexes, as illustrated later in this chapter. In physical-science fields such as climatology or remote sensing, factor scores obtained by principal components without rotation help to analyze large datasets. In these applications, principal components are called “empirical orthogonal functions.” The first empirical orthogonal function, or EOF1, equals the factor score for the first unrotated principal component. EOF2 is the score for the second principal component, and so forth.

Principal Factoring

The examples above involve principal component factoring, specified by the command **factor** with option **pcf**. Other **factor** options perform different kinds of factor analysis.

- pcf** Principal component factoring
- pf** Principal factoring (default)
- ipf** Principal factoring with iterated communalities
- ml** Maximum-likelihood factoring

Principal factoring extracts principal components from a modified correlation matrix, in which the main diagonal consists of communality estimates instead of 1's. The **factor** options **pf** and **ipf** both perform principal factoring. They differ in how communalities are estimated:

- pf** Communality estimates equal R^2 from regressing each variable on all the others.
- ipf** Iterative estimation of communalities.

Whereas principal component analysis focuses on explaining the variables' variance, principal factoring explains intervariable correlations.

We apply principal factoring with iterated communalities (**ipf**) to the planetary data:

```
. factor rings logdsun - logdense, ipf
```

(obs=9)

Factor analysis/correlation
 Method: iterated principal factors Number of obs = 9
 Rotation: (unrotated) Retained factors = 5
 Number of params = 15

Beware: solution is a Heywood case
 (i.e., invalid or boundary values of uniqueness)

Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	4.59663	3.46817	0.7903	0.7903
Factor2	1.12846	1.05107	0.1940	0.9843
Factor3	0.07739	0.06438	0.0133	0.9976
Factor4	0.01301	0.01176	0.0022	0.9998
Factor5	0.00125	0.00137	0.0002	1.0000
Factor6	-0.00012	.	-0.0000	1.0000

LR test: independent vs. saturated: chi2(15) = 100.49 Prob>chi2 = 0.0000

Factor loadings (pattern matrix) and unique variances

variable	Factor1	Factor2	Factor3	Factor4	Factor5
rings	0.9760	0.0665	0.1137	-0.0206	-0.0223
logdsun	0.6571	-0.6705	0.1411	0.0447	0.0082
lograd	0.9267	0.3700	-0.0450	0.0486	0.0166
logmoons	0.9674	-0.0107	0.0078	-0.0859	0.0160
logmass	0.8378	0.5458	0.0056	0.0282	-0.0071
logdense	-0.8460	0.4894	0.2059	-0.0061	0.0100

variable	Uniqueness
rings	0.0292
logdsun	0.0966
lograd	-0.0004
logmoons	0.0564
logmass	-0.0007
logdense	0.0022

In this example, Stata gives an ominous warning: “Beware: solution is a Heywood case.” Clicking on the linked Heywood case warning yields an explanation of the problem which, in this instance, reflects our unusually small sample ($n = 9$). For simplicity, we will continue the analysis anyway, but in research such a warning should caution us to rethink our approach.

Only the first two factors have eigenvalues above 1. With **pcf** or **pf** factoring, we can simply disregard minor factors. Using **ipf**, however, we must decide how many factors to retain, and then repeat the analysis asking for exactly that many factors. Here we will retain two factors:

```
. factor rings logdsun - logdense, ipf factor(2)
```

(obs=9)

Factor analysis/correlation
 Method: iterated principal factors Number of obs = 9
 Rotation: (unrotated) Retained factors = 2
 Number of params = 11

Beware: solution is a Heywood case
 (i.e., invalid or boundary values of uniqueness)

Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	4.57495	3.47412	0.8061	0.8061
Factor2	1.10083	1.07631	0.1940	1.0000
Factor3	0.02452	0.02013	0.0043	1.0043
Factor4	0.00439	0.00795	0.0008	1.0051
Factor5	-0.00356	0.02182	-0.0006	1.0045
Factor6	-0.02537	.	-0.0045	1.0000

LR test: independent vs. saturated: chi2(15) = 100.49 Prob>chi2 = 0.0000

Factor loadings (pattern matrix) and unique variances

variable	Factor1	Factor2	Uniqueness
rings	0.9747	0.0537	0.0470
logdsun	0.6533	-0.6731	0.1202
lograd	0.9282	0.3605	0.0086
logmoons	0.9685	-0.0228	0.0614
logmass	0.8430	0.5462	-0.0089
logdense	-0.8294	0.4649	0.0960

After **ipf** factor analysis, we could create composite variables using **rotate** and **predict** as before. Due to the Heywood-case problem, **ipf** factor scores here are less plausible than our earlier **pcf** results. As a research strategy, it often helps to repeat factor analyses using several different methods, and compare these looking for stable conclusions.

Maximum-Likelihood Factoring

Maximum-likelihood factoring, unlike Stata's other **factor** options, provides formal hypothesis tests that help in determining the appropriate number of factors. To obtain a single maximum-likelihood factor for the planetary data, type

```
. factor rings logdsun - logdense, ml nolog factor(1)
(obs=9)

Factor analysis/correlation  

  Method: maximum likelihood  

  Rotation: (unrotated)  

  Log likelihood = -42.32054  

Number of obs = 9  

Retained factors = 1  

Number of params = 6  

Schwarz's BIC = 97.8244  

(Akaike's) AIC = 96.6411

Factor
```

Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	4.47258	.	1.0000	1.0000

LR test: independent vs. saturated: chi2(15) = 100.49 Prob>chi2 = 0.0000
 LR test: 1 factor vs. saturated: chi2(9) = 51.73 Prob>chi2 = 0.0000

Factor loadings (pattern matrix) and unique variances

Variable	Factor1	Uniqueness
rings	0.9873	0.0254
logdsun	0.5922	0.6493
lograd	0.9365	0.1229
logmoons	0.9589	0.0805
logmass	0.8692	0.2445
logdense	-0.7715	0.4049

The **ml** output includes two likelihood-ratio χ^2 tests:

LR test: independent vs. saturated

This tests whether a no-factor (independent) model fits the observed correlation matrix significantly worse than a saturated or perfect-fit model. A low probability (here 0.0000, meaning $p < .00005$) indicates that a no-factor model is too simple.

LR test: 1 factor vs. saturated

This tests whether the current 1-factor model fits significantly worse than a saturated model. The low p -value here suggests that one factor is too simple, as well.

Perhaps a 2-factor model will do better:

```
. factor rings logdsun - logdense, ml nolog factor(2)
(obs=9)

Factor analysis/correlation                               Number of obs      =      9
Method: maximum likelihood                           Retained factors =       2
Rotation: (unrotated)                                Number of params =     11
                                                       Schwarz's BIC    =  36.6881
Log likelihood = -6.259338                           (Akaike's) AIC   =  34.5187

Beware: solution is a Heywood case
        (i.e., invalid or boundary values of uniqueness)
```

Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	3.64200	1.67115	0.6489	0.6489
Factor2	1.97085	.	0.3511	1.0000

LR test: independent vs. saturated: $\text{chi}^2(15) = 100.49 \text{ Prob} > \text{chi}^2 = 0.0000$
 LR test: 2 factors vs. saturated: $\text{chi}^2(4) = 6.72 \text{ Prob} > \text{chi}^2 = 0.1513$
 (tests formally not valid because a Heywood case was encountered)

Factor loadings (pattern matrix) and unique variances

Variable	Factor1	Factor2	Uniqueness
rings	0.8655	-0.4154	0.0783
logdsun	0.2092	-0.8559	0.2236
lograd	0.9844	-0.1753	0.0003
logmoons	0.8156	-0.4998	0.0850
logmass	0.9997	0.0264	0.0000
logdense	-0.4643	0.8857	0.0000

Now we find the following:

LR test: independent vs. saturated

The first test is unchanged; a no-factor model is too simple.

LR test: 2 factors vs. saturated

The 2-factor model is *not* significantly worse ($p = .1513$) than a perfect-fit model.

These tests suggest that two factors provide an adequate model.

Computational routines performing maximum-likelihood factor analysis often yield Heywood solutions, or unrealistic results such as negative variance or zero uniqueness. When this happens (as in the 2-factor **ml** example above), the χ^2 tests lack formal justification. Viewed descriptively, the tests can still provide informal guidance regarding the appropriate number of factors.

Cluster Analysis — 1

Cluster analysis encompasses a variety of methods that divide observations into groups or clusters, based on their dissimilarities across a number of variables. It is most often used as an exploratory approach, for developing empirical typologies, rather than as a means of testing pre-specified hypotheses. Indeed, there exists little formal theory to guide hypothesis testing for the common clustering methods. The number of choices available at each step in the analysis is daunting, and all the more so because they can lead to many different results. This section provides an entry point to beginning cluster analysis. We review some basic ideas and illustrate them through a simple example. The section that follows considers a somewhat larger example. Stata's *Multivariate Statistics Reference Manual* introduces and defines the full range of choices available. Everitt *et al.* (2001) cover topics in more detail, including helpful comparisons among the many cluster-analysis methods.

All clustering methods begin with some definition of dissimilarity (or similarity). Dissimilarity measures reflect the differentness or distance between two observations, across a specified set of variables. Generally, such measures are designed so that two identical observations have a dissimilarity of 0, and two maximally different observations have a dissimilarity of 1. Similarity measures reverse this scaling, so that identical observations have a similarity of 1. Stata's **cluster** options offer many choices of dissimilarity or similarity measures. For purposes of calculation, Stata internally transforms similarity to dissimilarity:

$$\text{dissimilarity} = 1 - \text{similarity}$$

The default dissimilarity measure for **averagelinkage**, **completelinkage**, **singlelinkage** and **waveragelinkage** is the Euclidean distance, option **measure(L2)**. This defines the distance between observations i and j as

$$\left\{ \sum_k (x_{ki} - x_{kj})^2 \right\}^{1/2}$$

where x_{ki} is the value of variable x_k for observation i , x_{kj} the value of x_k for observation j , and summation occurs over all the x variables considered. Other choices available for measuring the (dis)similarities between observations based on continuous variables include the squared Euclidean distance (**L2squared**, default for **centroidlinkage**, **medianlinkage** and **wardslinkage**), the absolute-value distance (**L1**), maximum-value distance (**Linfinity**) and correlation coefficient similarity measure (**correlation**). Choices for binary variables include simple

matching (**matching**), Jaccard binary similarity coefficient (**Jaccard**) and others. The **gower** option works with a mix of continuous and binary variables. Type **help measure option** for a complete list and detailed explanations of dissimilarity-measure options.

Clustering methods fall into two broad categories, *partition* and *hierarchical*. Partition methods break the observations into a pre-set number of nonoverlapping groups. We have two ways to do this:

cluster kmeans Kmeans cluster analysis

User specifies the number of clusters (K) to create. Stata then finds these through an iterative process, assigning observations to the group with the closest mean.

cluster kmedians Kmedians cluster analysis

Similar to Kmeans, but with medians.

Partition methods tend to be computationally simpler and faster than hierarchical methods. The necessity of declaring the exact number of clusters in advance is a disadvantage for exploratory work, however.

Hierarchical methods involve a process of small groups gradually fusing to form increasingly large ones. Stata takes an *agglomerative* approach in hierarchical cluster analysis: It starts out with each observation considered as its own separate group. The closest two groups are merged, and this process continues until a specified stopping-point is reached, or all observations belong to one group. A graphical display called a *dendrogram* or *tree diagram* visualizes hierarchical clustering results. Several choices exist for the *linkage method*, which specifies what should be compared between groups that contain more than one observation:

cluster singlelinkage Single linkage cluster analysis

Computes the dissimilarity between two groups as the dissimilarity between the least different pair of observations in the two groups. Although simple, this method has low resistance to outliers or measurement errors. Observations tend to join clusters one at a time, forming unbalanced, drawn-out groups in which members have little in common, but are linked by intermediate observations — a problem called *chaining*.

cluster averagelinkage Average linkage cluster analysis

Uses the average dissimilarity of observations in the two groups, yielding properties intermediate between single and complete linkage. Simulation studies report that this works well for many situations and is reasonably robust (see Everitt *et al.* 2001, and sources they cite). Commonly used in archaeology.

cluster completelinkage Complete linkage cluster analysis

Uses the least similar pair of observations in the two groups. Less sensitive to outliers than single linkage, but with the opposite tendency towards clumping many observations into tight, spatially compact clusters.

cluster waveragelinkage Weighted-average linkage cluster analysis

cluster medianlinkage Median linkage cluster analysis.

Weighted-average linkage and median linkage are variations on average linkage and centroid linkage, respectively. In both cases, the difference is in how groups of unequal size are treated when merged. In average linkage and centroid linkage, the number of elements of each group are factored into the computation, giving correspondingly larger influence to the larger group (because each observation carries the same weight). In weighted-average linkage and median linkage, the two groups are given equal weighting regardless of how many observations there are in each group. Median linkage, like centroid linkage, is subject to reversals.

cluster centroidlinkage Centroid linkage cluster analysis

Centroid linkage merges the groups whose means are closest (in contrast to average linkage which looks at the average distance between elements of the two groups). This method is subject to reversals — points where a fusion takes place at a lower level of dissimilarity than an earlier fusion. Reversals signal an unstable cluster structure, are difficult to interpret, and cannot be graphed by **cluster dendrogram**.

cluster wardslinkage Ward's linkage cluster analysis

Joins the two groups that result in the minimum increase in the error sum of squares. Does well with groups that are multivariate normal and of similar size, but poorly when clusters have unequal numbers of observations.

Earlier in this chapter, a principal component factor analysis of variables in *planets.dta* (Figure 11.3) identified three types of planets: inner rocky planets, outer gas giants and, in a class by itself, Pluto. Cluster analysis provides an alternative approach to the question of planet types. Because variables such as number of moons (*moons*) and mass in kilograms (*mass*) are measured in incomparable units, with hugely different variances, we should standardize in some way to avoid results dominated by the highest-variance items. A common, although not automatic, choice is standardization to zero mean and unit standard deviation. This is accomplished through the **egeen** command (and using variables in log form, for the same reasons discussed earlier). **summarize** confirms that the new *z* variables have (near) zero means, and standard deviations equal to one.

```
. egen zrings = std(rings)
. egen zlogdsun = std(logdsun)
. egen zloggrad = std(loggrad)
. egen zlogmoon = std(logmoons)
. egen zlogmass = std(logmass)
. egen zlogdens = std(logdense)
. summ z*
```

Variable	Obs	Mean	Std. Dev.	Min	Max
zrings	9	-1.99e-08	1	-.8432741	1.054093
zlogdsun	9	-1.16e-08	1	-1.393821	1.288216
zloggrad	9	-3.31e-09	1	-1.3471	1.372751
zlogmoon	9	0	1	-1.207296	1.175849
zlogmass	9	-4.14e-09	1	-1.74466	1.365167
zlogdens	9	-1.32e-08	1	-1.453143	1.128901

The “three types” conclusion suggested by our principal component factor analysis is robust, and could have been found through cluster analysis as well. For example, we might perform a

hierarchical cluster analysis with average linkage, using Euclidean distance (**L2**) as our dissimilarity measure. The option **name(L2avg)** gives the results from this particular analysis a name, so that we can refer to them in later commands. The results-naming feature is convenient when we need to try a number of cluster analyses and compare their outcomes.

```
. cluster averagelinkage zrings zlogdsun zlograd zlogmoon zlogmass
zlogdens, measure(L2) name(L2avg)
```

Nothing seems to happen, although we might notice that our dataset now contains three new variables with names based on *L2avg*. These new *L2avg** variables are not directly of interest, but can be used unobtrusively by the **cluster dendrogram** command to draw a cluster analysis tree or dendrogram visualizing the most recent hierarchical cluster analysis results (Figure 11.4). The **label(planet)** option here causes planet names (values of *planet*) to appear as labels below the graph.

```
. cluster dendrogram, label(planet) ylabel(0(1)5)
```

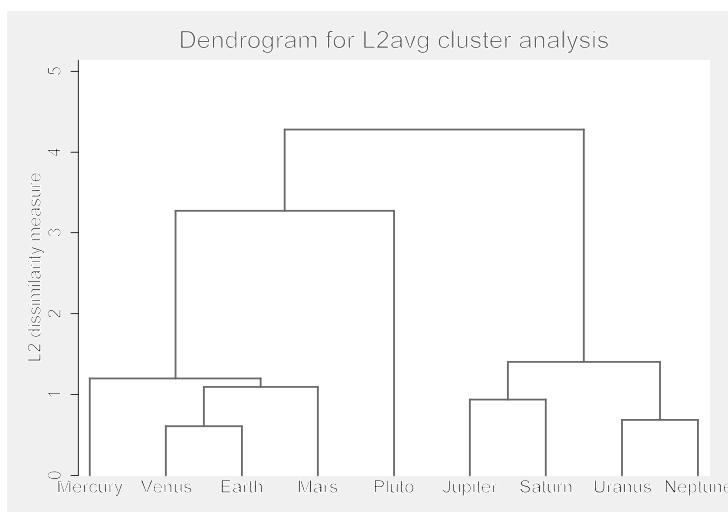


Figure 11.4

Dendograms such as Figure 11.4 provide key interpretive tools for hierarchical cluster analysis. We can trace the agglomerative process from each observation comprising its own unique cluster, at bottom, to all fused into one cluster, at top. Venus and Earth, and also Uranus and Neptune, are the least dissimilar or most alike pairs. They are fused first, forming the first two multi-observation clusters at a height (dissimilarity) below 1. Jupiter and Saturn, then Venus–Earth and Mars, then Venus–Earth–Mars and Mercury, and finally Jupiter–Saturn and Uranus–Neptune are fused in quick succession, all with dissimilarities slightly above 1. At this point we have the same three groups suggested in Figure 11.3 by principal components: the inner rocky planets, the gas giants and Pluto. The three clusters remain stable until, at much higher dissimilarity (above 3), Pluto fuses with the inner rocky planets. At a dissimilarity above 4, the final two clusters fuse.

So, how many types of planets are there? Figure 11.4 makes clear that the answer is “it depends.” How much dissimilarity do we want to accept within each type? The long vertical lines between the three-cluster stage and the two-cluster stage in the upper part of the graph indicate that we have three fairly distinct types. We could reduce this to two types only by fusing an observation (Pluto) that is quite dissimilar to others in its group. We could expand it to five types only by drawing distinctions between several planets (e.g., Mercury and Venus–Earth–Mars) that by solar-system standards are not greatly dissimilar. Thus, the dendrogram makes a case for a three-type scheme.

The **cluster generate** command creates a new variable indicating the type or group to which each observation belongs. In this example, **groups(3)** calls for three groups. The **name(L2avg)** option specifies the particular results we named *L2avg*. This option is most useful when our session included multiple cluster analyses.

```
. cluster generate plantype = groups(3), name(L2avg)
. label variable plantype "Planet type"
. list planet plantype
```

	planet	plantype
1.	Mercury	1
2.	Venus	1
3.	Earth	1
4.	Mars	1
5.	Jupiter	3
6.	Saturn	3
7.	Uranus	3
8.	Neptune	3
9.	Pluto	2

The inner rocky planets have been coded as *plantype* = 1, the gas giants as *plantype* = 3, and Pluto is by itself as *plantype* = 2. The group designations as 1, 2 and 3 follow the left-to-right ordering of final clusters in the dendrogram (Figure 11.4). Once the data have been saved, our new typology could be used like any other categorical variable in subsequent analyses.

These planetary data have a strong pattern of natural groups, which is why such different techniques as cluster analysis and principal components point towards similar conclusions. We could have chosen other dissimilarity measures and linkage methods for this example, and still arrived at much the same place. Complex or weakly patterned data, on the other hand, often yield quite different results depending on the methods used. The clusters found by one method might not prove replicable under other methods, or with slightly different analytical decisions.

Cluster Analysis — 2

Discovering a simple, robust typology to describe nine planets was straightforward. For a more challenging example, consider the cross-national data in *Nations2.dta*. These United Nations human-development variables could be applied to develop an empirical typology of nations.

```
. use C:\data\Nations2.dta, clear
```

. describe				
Contains data from c:\data\Nations2.dta				UN Human Development Indicators
obs:	194 <th>vars:</th> <td>13</td> <th>12 Jun 2012 12:43</th>	vars:	13	12 Jun 2012 12:43
size:	12,804			
variable	storage type	display format	value label	variable label
country	str21	%21s		Country
region	byte	%8.0g		Region
gdp	float	%9.0g	region	Gross domestic product per cap 2005\$, 2006/2009
school	float	%9.0g		Mean years schooling (adults) 2005/2010
adfert	float	%8.0g		Adolescent fertility: births/1000 fem 15-19, 2010
chldmort	float	%9.0g		Prob dying before age 5/1000 live births 2005/2009
life	float	%9.0g		Life expectancy at birth 2005/2010
pop	float	%9.0g		Population 2005/2010
urban	float	%9.0g		Percent population urban 2005/2010
femlab	float	%9.0g		Female/male ratio in labor force 2005/2009
literacy	float	%9.0g		Adult literacy rate 2005/2009
co2	float	%9.0g		Tons of CO2 emitted per cap 2005/2006
gini	float	%9.0g		Gini coef income inequality 2005/2009

Sorted by: region country

Working with the same data in Chapter 7, we saw that nonlinear transformations such as logarithms helped to normalize distributions and linearize relationships among some of these variables. Similar arguments for nonlinear transformations could apply to cluster analysis, but to keep the example simple they are not pursued here. Linear transformations to standardize the variables in some fashion remain essential, however. Otherwise, the variable *gdp*, which ranges from about \$280 to \$74,906 (standard deviation \$13,942) would overwhelm other variables such as *life*, which ranges from about 46 to 83 years (standard deviation 10 years). In the previous section, we standardized planetary data by subtracting each variable's mean, then dividing by its standard deviation, so that the resulting z-scores all had standard deviations of one. In this section we take a different approach, range standardization, which also works well for cluster analysis.

Range standardization involves dividing each variable by its range. There is no command to do this directly in Stata, but we can improvise one easily enough. To do so, we make use of results that Stata stores unobtrusively after **summarize**. Recall that we can see a complete list of stored results after **summarize** by typing the command **return list**. (After modeling procedures such as **regress** or **factor**, use the command **ereturn list** instead.) In this example, we view the results stored after **summarize pop**, then use the maximum and minimum values (stored as scalars that Stata calls *r(max)* and *r(min)*) to calculate a new, range-standardized version of population.

. summarize gdp

Variable	Obs	Mean	Std. Dev.	Min	Max
gdp	179	12118.74	13942.34	279.8	74906

```
. return list
scalars:
r(N) = 179
r(sum_w) = 179
r(mean) = 12118.73919336756
r(Var) = 194388878.6050418
r(sd) = 13942.34121677711
r(min) = 279.7999877929688
r(max) = 74906
r(sum) = 2169254.315612793

. generate rgdp = gdp/(r(max) - r(min))
. label variable rgdp "Range-standardized GDP"
```

Similar commands create range-standardized versions of other living-conditions variables:

```
. quietly summ school
. generate rschool = school/(r(max) - r(min))
. label variable rschool "Range-standardized schooling"
. quietly summ adfert
. generate radfert = adfert/(r(max) - r(min))
. label variable radfert "Range-standardized adolescent fertility"
```

and so forth, defining the 8 new variables listed below.

```
. describe rgdp - rfemlab
```

variable	name	storage type	display format	value label	variable label
rgdp		float	%9.0g		Range-standardized GDP
rschool		float	%9.0g		Range-standardized schooling
radfert		float	%9.0g		Range-standardized adolescent fertility
rchldmort		float	%9.0g		Range-standardized child mortality
rlife		float	%9.0g		Range-standardized life expectancy
rpop		float	%9.0g		Range-standardized population
rurban		float	%9.0g		Range-standardized percent urban
rfemlab		float	%9.0g		Range-standardized female labor ratio

If our **generate** commands were done correctly, these new range-standardized variables should all have ranges equal to 1. **tabstat** confirms that they do.

```
. tabstat rgdp - rfemlab, statistics(range)
stats | rgdp rschool radfert rchldm~t rlife rpop
range | 1 .9999999 1 1 1 1
stats | rurban rfemlab
range | .9999999 1
```

Once the variables of interest have been standardized, we can proceed with cluster analysis. As we divide more than 100 nations into types, we have no reason to assume that each type will include a similar number of nations. Average linkage (used in our planetary example), along

with some other methods, gives each observation the same weight. This tends to make larger clusters more influential as agglomeration proceeds. Weighted average and median linkage methods, on the other hand, give equal weight to each cluster regardless of how many observations it contains. Such methods consequently tend to work better for detecting clusters of unequal size. Median linkage, like centroid linkage, is subject to reversals (which will occur with these data), so the following example applies weighted average linkage. Absolute-value distance (**measure(L1)**) provides our dissimilarity measure.

```
. cluster wavagelinkage rgdp - rfemlab, measure(L1) name(L1wav)
```

The full cluster analysis proves unmanageably large for a dendrogram:

```
. cluster dendrogram
too many leaves; consider using the cutvalue() or cutnumber() options
r(198);
```

Following the error-message advice, Figure 11.5 employs a **cutnumber(100)** option to form a dendrogram that starts with only 100 groups, after the first few fusions have taken place.

```
. cluster dendrogram, ylabel(0(.5)3) cutnumber(100)
```

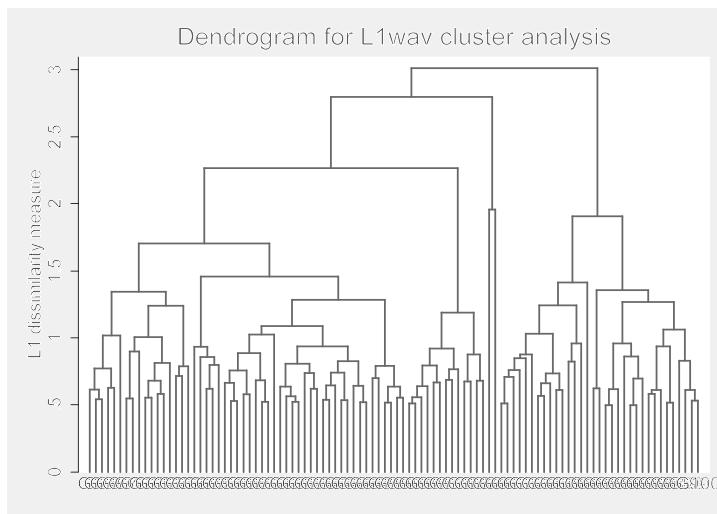


Figure 11.5

The bottom labels in Figure 11.5 are unreadable, but we can trace the general flow of this clustering process. Most of the fusion takes place at dissimilarities below 1. Two nations just right of center in the dendrogram are unusual; they resist fusion until about 2 and then form a stable two-nation group quite different from all the rest. This is one of four clusters remaining at dissimilarities above 2. The first and fourth of these four final clusters (reading left to right) appear heterogeneous, formed through successive fusion of a number of somewhat distinct major subgroups. The second cluster, in contrast, appears more homogeneous. It combines many

nations that fused into two subgroups at dissimilarities below 1, and then fused into one group at slightly above 1.

Figure 11.6 gives another view of this analysis, this time using the **cutvalue(1.2)** option to show only clusters with dissimilarities above 1.2, after most of the fusions have taken place. The **showcount** option calls for bottom labels ($n=13, n=11, \dots$) that indicate the number of nations in each group. We see that groups 8, 9 and 12 each consists of a single nation.

```
. cluster dendrogram, ylabel(0(.5)3) cutvalue(1.2) showcount
```

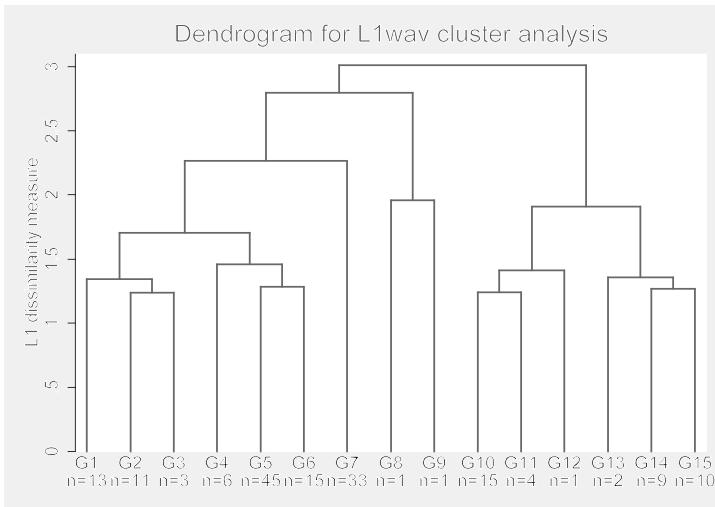


Figure 11.6

As Figure 11.6 shows, there are 15 groups remaining at dissimilarities above 1.2. For purposes of illustration, we will consider only the top four groups, which have dissimilarities above 2. **cluster generate** creates a categorical variable for the final four groups from the cluster analysis we named *L1wav*.

```
. cluster generate ctype = groups(4), name(L1wav)
. label variable ctype "Country type"
```

We could next examine which countries belong to which groups by typing

```
. sort ctype
. by ctype: list country
```

The long list that results is not shown here, but it reveals that the two-nation cluster we noticed in Figure 11.5 is type 3, India and China. The relatively homogeneous second cluster in Figure 11.5, now classified as type 4, contains a large group of the poorest nations, mainly in Africa. The relatively diverse type 2 contains more affluent nations including the U.S., Japan and many in Europe. Type 1, also diverse, contains nations with intermediate living conditions. Whether this or some other typology is meaningful remains a substantive question, not a statistical one,

and depends on the uses for which the typology is needed. Choosing different options in the steps of our cluster analysis would have returned different results. By experimenting with a variety of reasonable choices, we could gain a sense of which findings are most stable.

Using Factor Scores in Regression

Principal components and factor analysis often help to define new composite variables for further analysis. For example, the factor scores calculated by **predict** could become independent or dependent variables in subsequent regression analyses. To illustrate this process we turn to the survey dataset *PNWsurvey2_11.dta*.

Contains data from c:\data\PNWsurvey2_11.dta				
obs:	734	Pacific NW CERA survey (February 2011)		
vars:	16	13 Jun 2012 16:11		
size:	13,946			
variable name	storage type	display format	value label	variable label
age	byte	%8.0g	age	Age in years
sex	byte	%8.0g	sex	Gender
educ	byte	%14.0g	degree	Highest degree completed
party	byte	%11.0g	party	Political party identification
newcomer	byte	%9.0g	yesno	Moved here within past 5 years
surveytwt	float	%9.0g		CERA survey
				wt--adults/age/race/sex/county
forest cutting	byte	%13.0g	eff	Loss of forestry jobs or income
	byte	%13.0g	eff	Overharvesting or heavy cutting of timber
losfish	byte	%13.0g	eff	Loss of fishing jobs or income
overfish	byte	%13.0g	eff	Overfishing in the ocean
water1	byte	%13.0g	eff	Water quality or pollution issues
water2	byte	%13.0g	eff	Water supply problems
warming	byte	%13.0g	eff	Global warming or climate change
sprawl	byte	%13.0g	eff	Urban sprawl/development of countryside
weather	byte	%13.0g	eff	Unusual/extreme weather-related events
losscen	byte	%13.0g	eff	Loss of scenic natural beauty

Sorted by:

The 16 variables in this dataset represent a subset from a telephone survey of residents in a coastal region of the Pacific Northwest. This survey, conducted in February 2011, formed part of a series of regional surveys under the Community and Environment in Rural America (CERA) initiative (e.g., Hamilton et al. 2010b; Safford and Hamilton 2010).

Ten of the questions for this example, *forest* through *losscen*, ask whether particular environmental issues have had local effects. The questions go as follows.

I'm going to read a list of environmental issues that might be problems in some rural places. With regard to the place where YOU live, for each issue I'd like to know whether you think this has had no effect, had minor effects, or had major effects ON YOUR FAMILY OR COMMUNITY OVER THE PAST 5 YEARS?

Loss of forestry jobs or income?
Overharvesting or heavy cutting of timber?
Loss of fishing jobs or income?
Overfishing in the ocean?
Water quality and pollution issues?
Water supply problems?
Global warming or climate change?
Urban sprawl or rapid development of the countryside?
Unusual or extreme weather-related events?
Loss of scenic natural beauty?

For each question, respondents can say whether that issue had no effect, minor effects or major effects on their family or community.

```
. svy: tab forest
(running tabulate on estimation sample)
```

Number of strata	=	1	Number of obs	=	734
Number of PSUs	=	734	Population size	=	725.97798
			Design df	=	733

Loss of forestry jobs or income	proportions
None	.2084
Minor	.2108
Major	.5808
Total	1

Key: proportions = cell proportions

Many respondents reported that loss of forestry or fishing jobs had major effects. Water supplies, urban sprawl and loss of scenic beauty were less immediate concerns for their region, as depicted in Figure 11.7. This overview graphic was created by drawing ten simple bar charts using **catplot** with analytical weights given by the *surveywt* variable ([aw = *surveywt*], see Chapter 4), then combining them into a single image with **graph combine**.



Figure 11.7

Does concern about these 10 environmental issues reflect a smaller number of underlying dimensions? Principal factoring with iterated communalities (**ipf**) identifies the underlying dimensions that best account for the pattern of correlations between variables.

```
. factor forest cutting losfish overfish water1 water2
   warming sprawl weather losscen, ipf

(obs=734)

Factor analysis/correlation
  Method: iterated principal factors
  Rotation: (unrotated)                               Number of obs      =      734
                                                       Retained factors =         9
                                                       Number of params =        45



| Factor   | Eigenvalue | Difference | Proportion | Cumulative |
|----------|------------|------------|------------|------------|
| Factor1  | 3.34457    | 2.18016    | 0.5886     | 0.5886     |
| Factor2  | 1.16440    | 0.78239    | 0.2049     | 0.7936     |
| Factor3  | 0.38201    | 0.06367    | 0.0672     | 0.8608     |
| Factor4  | 0.31834    | 0.09420    | 0.0560     | 0.9168     |
| Factor5  | 0.22413    | 0.06330    | 0.0394     | 0.9563     |
| Factor6  | 0.16083    | 0.11590    | 0.0283     | 0.9846     |
| Factor7  | 0.04494    | 0.00940    | 0.0079     | 0.9925     |
| Factor8  | 0.03554    | 0.02822    | 0.0063     | 0.9988     |
| Factor9  | 0.00732    | 0.00757    | 0.0013     | 1.0000     |
| Factor10 | -0.00025   | .          | -0.0000    | 1.0000     |



LR test: independent vs. saturated: chi2(45) = 2030.48 Prob>chi2 = 0.0000


```

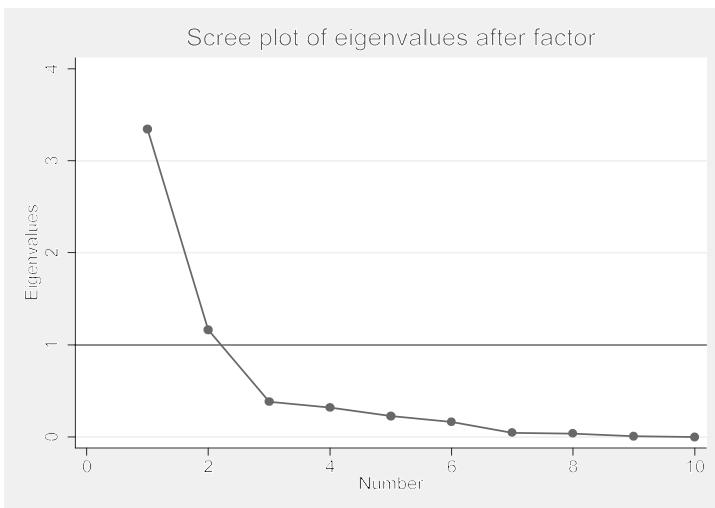
Factor loadings (pattern matrix) and unique variances

Variable	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
forest	0.3408	0.6168	0.0569	0.3357	-0.0288	0.0285
cutting	0.6254	0.1541	0.0641	-0.0126	-0.3822	-0.0466
losfish	0.4637	0.6746	-0.0195	-0.0834	0.2061	0.0068
overfish	0.6568	0.2032	-0.0637	-0.3887	-0.0030	-0.0670
water1	0.6410	-0.1694	-0.2377	0.0438	-0.0795	0.2092
water2	0.6181	-0.2101	-0.3671	0.1051	0.0989	0.0258
warming	0.6393	-0.2047	0.2271	-0.0950	0.0374	0.1280
sprawl	0.6046	-0.2189	-0.0078	0.1117	0.0683	-0.1921
weather	0.4960	-0.1952	0.3432	0.0485	0.1012	0.1310
losscen	0.6144	-0.2511	0.0976	0.1033	0.0483	-0.1960

Variable	Factor7	Factor8	Factor9	Uniqueness
forest	0.0608	0.0347	0.0045	0.3809
cutting	0.0004	-0.0223	-0.0035	0.4321
losfish	-0.0727	-0.0585	-0.0069	0.2713
overfish	0.0416	0.0818	0.0049	0.3593
water1	-0.1172	0.0083	0.0041	0.4381
water2	0.1055	0.0078	-0.0146	0.4061
warming	0.0772	-0.1150	0.0029	0.4518
sprawl	-0.0294	-0.0163	0.0625	0.5273
weather	-0.0129	0.1005	0.0029	0.5580
losscen	-0.0486	-0.0017	-0.0554	0.4930

How many factors should we retain? Only the first two have eigenvalues above 1, although with principal factoring (unlike principal components) an eigenvalue-1 cutoff is sometimes viewed as too strict. A scree plot (Figure 11.8), however, visually confirms that after the first two factors, the remainder all have similar, low eigenvalues.

```
. screeplot
```

**Figure 11.8**

After further experiments with retaining and rotating two, three or more factors (not shown), it appears that two factors provide the most interpretable results — an important consideration. Once we decide to proceed with two factors, the next first step (because we are using iterated principal factoring) involves repeating the analysis with a **factor(2)** restriction.

. factor forest cutting losfish overfish water1 water2 warming sprawl weather losscen, ipf factor(2)				
(obs=734)				
Factor analysis/correlation				
Method: iterated principal factors				
Rotation: (unrotated)				
Number of obs = 734				
Retained factors = 2				
Number of params = 19				
Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	3.20673	2.14023	0.7504	0.7504
Factor2	1.06650	0.85629	0.2496	1.0000
Factor3	0.21021	0.08664	0.0492	1.0492
Factor4	0.12357	0.03148	0.0289	1.0781
Factor5	0.09209	0.06060	0.0215	1.0997
Factor6	0.03149	0.08506	0.0074	1.1070
Factor7	-0.05357	0.05634	-0.0125	1.0945
Factor8	-0.10991	0.00827	-0.0257	1.0688
Factor9	-0.11817	0.05758	-0.0277	1.0411
Factor10	-0.17575	.	-0.0411	1.0000
LR test: independent vs. saturated: chi2(45) = 2030.48 Prob>chi2 = 0.0000				
Factor loadings (pattern matrix) and unique variances				
Variable	Factor1	Factor2	Uniqueness	
forest	0.3217	0.5026	0.6439	
cutting	0.6020	0.1239	0.6223	
losfish	0.4788	0.7268	0.2425	
overfish	0.6289	0.1824	0.5713	
water1	0.6249	-0.1549	0.5855	
water2	0.5916	-0.1811	0.6172	
warming	0.6305	-0.1924	0.5654	
sprawl	0.6074	-0.2205	0.5824	
weather	0.4799	-0.1770	0.7384	
losscen	0.6155	-0.2515	0.5579	

Promax (oblique) rotation simplifies factor patterns while permitting some degree of correlation between the factors. Correlated factors will be statistically less parsimonious, because they have overlapping variance. However, they may be more realistic, if we view these factors as representing basic, not necessarily unrelated dimensions of environmental concern.

```
. rotate, promax
```

Factor analysis/correlation
 Method: iterated principal factors
 Rotation: oblique promax (Kaiser off)

Number of obs =	734
Retained factors =	2
Number of params =	19

Factor	Variance	Proportion	Rotated factors are correlated
Factor1	3.06450	0.7171	
Factor2	1.86514	0.4365	

LR test: independent vs. saturated: chi2(45) = 2030.48 Prob>chi2 = 0.0000

Rotated factor loadings (pattern matrix) and unique variances

Variable	Factor1	Factor2	Uniqueness
forest	-0.0567	0.6164	0.6439
cutting	0.4346	0.2980	0.6223
losfish	-0.0703	0.8951	0.2425
overfish	0.4190	0.3668	0.5713
water1	0.6389	0.0124	0.5855
water2	0.6277	-0.0244	0.6172
warming	0.6686	-0.0253	0.5654
sprawl	0.6674	-0.0612	0.5824
weather	0.5291	-0.0513	0.7384
losscen	0.6949	-0.0915	0.5579

Factor rotation matrix

	Factor1	Factor2
Factor1	0.9662	0.6109
Factor2	-0.2578	0.7917

The survey questions about local effects of water quality or supply issues, global warming, extreme weather, urban sprawl and loss of scenic beauty all load much higher on factor 1. Concern about loss of forestry or fishing jobs load much higher on factor 2. Two items that involve both resource overuse and jobs, *cutting* and *overfish*, have roughly similar loadings on both factors. Based on these patterns we could interpret factor 1 as representing general environmental concern, and factor 2 as representing resource jobs concern. **predict** calculates factor scores, which are composite variables defined as sums of the variables' standardized values, weighted by their factor score coefficients. The two new composite variables are named *enviro* and *resjobs*.

```
. predict enviro resjobs
(regression scoring assumed)
```

```
Scoring coefficients (method = regression; based on promax(3) rotated
factors)
```

Variable	Factor1	Factor2
forest	0.00448	0.15784
cutting	0.12496	0.14577
losfish	0.01218	0.68645
overfish	0.13504	0.09935
water1	0.18799	0.02717
water2	0.16724	0.00984
warming	0.20531	0.00398
sprawl	0.19251	-0.00430
weather	0.11661	0.00008
losscen	0.21054	-0.01245

```
. label variable enviro "Pollution, sprawl and scenic effects"
. label variable resjobs "Resource job loss effects"
. summ enviro resjobs
```

Variable	Obs	Mean	Std. Dev.	Min	Max
enviro	734	-1.38e-09	.9112633	-1.307359	2.209436
resjobs	734	9.40e-11	.9039373	-1.912148	.9766908

Social scientists constructing new composite variables look for evidence of their validity, or relation to what we think they measure. In this example one type of validity, face validity, is supported by the interpretable pattern of factor loadings. A second type, criterion validity, can be explored by testing whether the new variables correlate with other variables as theory or previous research predicts they should. For example, the single most robust finding from recent survey research on environmental concern in the U.S. has been the prevalence of ideological or political party-line effects. Box plots showing the distribution of scores on our general environmental-concern factor (*enviro*) over respondents' political *party* clearly follow this expected pattern. A horizontal line in Figure 11.7 marks the overall median, taken from result **r(p50)** of a **summarize, detail** command with survey sampling weights used as analytical weights. Among respondents who identify themselves as Republican, those with high *enviro* scores appear as outliers.

```
. quietly summ enviro, detail
. graph box enviro [aw = surveywt], over(party) yline(`r(p50)')
```

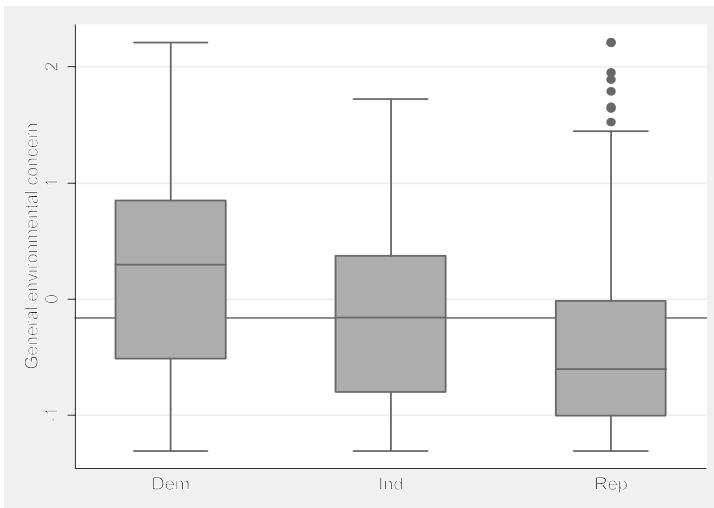


Figure 11.9

Other common findings reported by previous studies include effects from age, gender and education. The CERA researchers also had an interest in whether the environmental perceptions of rural-area newcomers differ from those of long-term residents. The regression below finds significant education, party and newcomer effects. Respondents who have more education, are Democrats or Independents, and have lived in this region for more than 5 years more often perceive local impacts from these environmental problems.

```
. svy: regress enviro age-newcomer
(running regress on estimation sample)

Survey: Linear regression

Number of strata      =           1
Number of PSUs        =       734
Number of obs          =       734
Population size        = 725.97798
Design df              =       733
F( 5, 729)            =      15.64
Prob > F              =     0.0000
R-squared              =     0.1275
```

enviro	Coef.	Linearized Std. Err.	t	P> t	[95% Conf. Interval]
age	-.0008836	.0027041	-0.33	0.744	-.0061923 .004425
sex	.1037294	.0807127	1.29	0.199	-.0547262 .2621849
educ	.1037535	.0383315	2.71	0.007	.0285008 .1790062
party	-.2949115	.0430073	-6.86	0.000	-.3793436 -.2104794
newcomer	-.2197846	.1089288	-2.02	0.044	-.4336341 -.005935
_cons	.2841083	.2264291	1.25	0.210	-.1604185 .7286351

A similar regression with our resource-jobs concern factor *resjobs* finds that it is less affected by education or politics. Instead, age and newcomer status are the strongest predictors. Younger

respondents, and those who moved into this area within the past five years, less often perceive impacts from the loss of fishing and forestry jobs.

```
. svy: regress resjobs age-newcomer
(running regress on estimation sample)

Survey: Linear regression

Number of strata = 1
Number of PSUs = 734
Number of obs = 734
Population size = 725.97798
Design df = 733
F( 5, 729) = 6.02
Prob > F = 0.0000
R-squared = 0.0792
```

resjobs	Coef.	Linearized Std. Err.	t	P> t	[95% Conf. Interval]
age	.008514	.0031023	2.74	0.006	.0024235 .0146045
sex	.0826771	.0883132	0.94	0.349	-.0906998 .2560541
educ	.0613762	.0414013	1.48	0.139	-.0199031 .1426555
party	-.0835506	.0479571	-1.74	0.082	-.1777002 .010599
newcomer	-.3624841	.1330725	-2.72	0.007	-.6237327 -.1012354
_cons	-.488574	.2391453	-2.04	0.041	-.9580654 -.0190826

Measurement and Structural Equation Models

Chapter 8 took a first look at structural equation modeling, beginning with a regression-like example involving relationships among observed variables (Figure 8.15). Structural equation models can also incorporate measurement models, which resemble factor analysis. Measurement models posit one or more unobserved, factor-like latent variables that cause variation in observed variables. Figure 11.10 illustrates using the Pacific Northwest CERA survey.

```
. use C:\data\PNWsurvey2_11.dta, clear
```

Two unobserved, latent variables named *Enviro* and *Resjobs* appear in Figure 11.10. Their names intentionally echo those given to factor scores in the previous section, but for this analysis we are starting over. No variables by those names exist. In structural equation modeling, Stata (by default) follows a convention of representing latent variables with names that start with a capital letter. Latent variable *Enviro* is depicted as causing some of the variation in six observed variables, the same ones that loaded mainly on our factor 1 in the previous section. Latent variable *Resjobs* explains variation in four other observed variables which loaded mainly or in part on factor 2. Note the graphical conventions of oval outlines for latent variables and rectangular outlines for observed variables. A curved, double-headed arrow represents the non-causal correlation between *Enviro* and *Resjobs*.

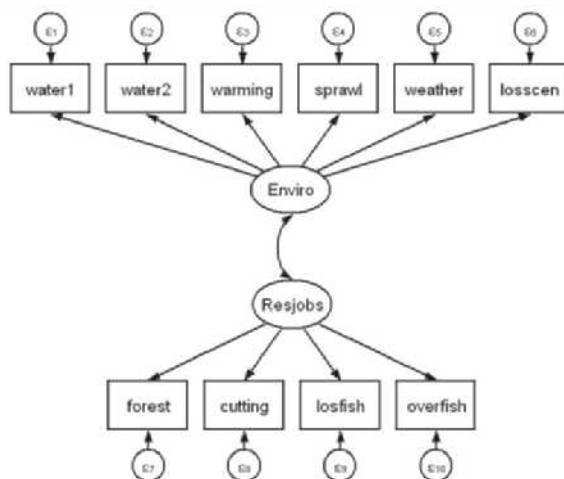
**Figure 11.10**

Figure 11.10 was drawn using Stata's SEM Builder, through the following steps.

Statistics > SEM (structural equation modeling) > Model building and estimation

Select the Add measurement Component (M) tool from the left-hand margin, and place it at the location desired for a latent variable within the plot space. Give the latent variable a name starting with a capital letter, such as *Enviro*. Select Measurement variables to be explained by this latent variable by choosing their names from a list. Select a Measurement direction, such as Up. Click OK, then repeat for a second latent variable in a different location such as *Resjobs*, with Measurement direction ... Down. Click OK again. Finally, use the Add covariance (C) tool from the right margin to place a curved, double-headed arrow for the correlation or covariance between the latent variables.

Figure 11.11 shows the same measurement model after estimation. Coefficients along the paths from latent to observed variables represent standardized regression coefficients analogous to factor loadings. Each observed variable also has its own unique variance, given by the ϵ (epsilon) terms in this path model. *Enviro* and *Resjobs* have a correlation of .62.

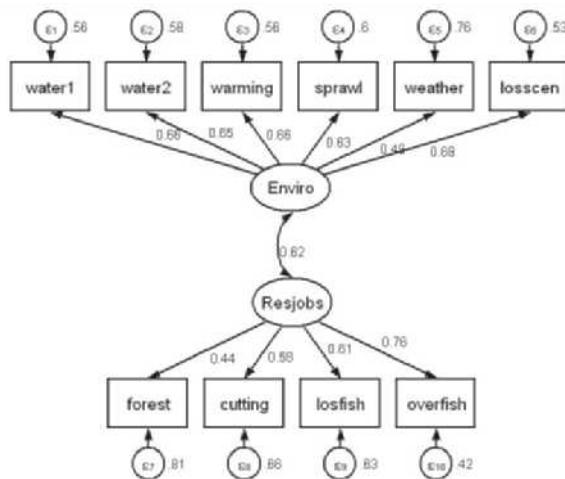


Figure 11.11

Once a path model such as Figure 11.10 has been drawn, simply clicking on Estimate will populate the graph with statistical results. Stata displays abundant information by default, but for some purposes, such as publication, we might want to keep things simple. Figure 11.11 contains only survey-weighted, standardized regression coefficients or variances, all in fixed display format with two digits right of the decimal. Simplifications were controlled through the following menu choices.

Settings > Variables > All ... > Results > Exogenous variables > None > OK
 Settings > Variables > All ... > Results > Endogenous variables > None > OK
 Settings > Variables > Error ... > Results > Error std. variance > OK
 Settings > Connections > Paths > Results > Std. parameter > OK
 Settings > Connections > All > Results > Result 1 > Format %3.2f > OK > OK
 Estimation > Estimate > Weights > Sampling weights ... (select variable such as *surveywt*) > OK

The same model can be estimated directly by a **sem** command, without going through the SEM Builder. In the command below, note the **svy:** prefix, which applies survey weighting to **sem** as it does to **regress** and many other Stata procedures. Standardized coefficients from this output correspond to the path coefficients in Figure 11.11. Unique variances of the observed variables, and the standardized covariance (i.e., correlation) between latent variables, likewise correspond to values shown in Figure 11.11.

```
. svy: sem (Enviro -> water1 water2 warming sprawl weather losscen)
          (Resjobs -> forest cutting losfish overfish), standard
```

Survey: Structural equation model

Number of strata = 1	Number of obs = 734
Number of PSUs = 734	Population size = 725.97798
	Design df = 733

{ 1) [water1]Enviro = 1
{ 2) [forest]Resjobs = 1

Standardized	Coef.	Linearized Std. Err.	t	P> t	[95% Conf. Interval]
Measurement					
water1 <- Enviro _cons	.6617216 .9865716	.0305711 .0405518	21.65 24.33	0.000 0.000	.6017043 .9069601 .7217389 1.066183
water2 <- Enviro _cons	.6474885 .6092422	.033683 .0301459	19.22 20.21	0.000 0.000	.5813618 .5500596 .7136151 .6684249
warming <- Enviro _cons	.6604354 .9570574	.0295415 .0398204	22.36 24.03	0.000 0.000	.6024393 .8788818 .7184315 1.035233
sprawl <- Enviro _cons	.6331502 .8070991	.0394133 .0351607	16.06 22.95	0.000 0.000	.5557738 .7380715 .7105265 .8761267
weather <- Enviro _cons	.4872326 1.229424	.0408934 .0493018	11.91 24.94	0.000 0.000	.4069505 1.132634 .5675148 1.326213
losscen <- Enviro _cons	.6868967 .8308351	.0336976 .0357022	20.38 23.27	0.000 0.000	.6207413 .7607443 .7530521 .9009259
forest <- Resjobs _cons	.4351381 1.701551	.0782853 .0851467	5.56 19.98	0.000 0.000	.2814481 1.534391 .5888282 1.868712
cutting <- Resjobs _cons	.5800483 1.086461	.0483413 .0455767	12.00 23.84	0.000 0.000	.4851445 .9969847 .6749521 1.175938
losfish <- Resjobs _cons	.611518 1.699434	.0646856 .0899735	9.45 18.89	0.000 0.000	.4845268 1.522798 .7385092 1.876071
overf~h <- Resjobs _cons	.7640356 1.211127	.0396915 .0531622	19.25 22.78	0.000 0.000	.686113 1.106759 .8419582 1.315495
Variance					
e.water1	.5621246	.0404591			.4880516 .6474398
e.water2	.5807587	.0436187			.5011403 .6730265
e.warming	.563825	.0390206			.4921959 .6458784
e.sprawl	.5991209	.049909			.5087318 .7055699
e.weather	.7626044	.0398492			.6882511 .8449901
e.losscen	.5281729	.0462936			.4446786 .6273445
e.forest	.8106548	.0681298			.6873535 .9560745
e.cutting	.663544	.0560805			.5620954 .7833023
e.losfish	.6260458	.0791129			.4884978 .8023236
e.overfish	.4162496	.0606515			.3126948 .5540985
Enviro	1	:			:
Resjobs	1	:			:

Covariance						
Enviro Resjobs	.6194808	.0603082	10.27	0.000	.5010833	.7378782

Figure 11.12 shows an example that combines a measurement model (latent variable *Enviro*, explaining variation in the six observed variables *water1* through *losscen*) with a regression model in which *Enviro* itself is explained by five observed background variables, *age* through *newcomer*. Conceptually, Figure 11.12 resembles our analysis in the previous section taking factor score *enviro* as a dependent variable regressed on *age* through *newcomer*. The structural equation approach formally combines aspects of factor analysis and regression into one model. It supports estimation and testing of a wide range of alternative specifications such as error correlations, and relationships involving other observed or latent variables.

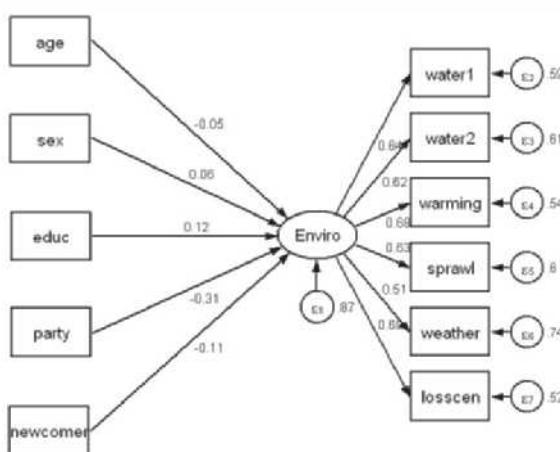


Figure 11.12

The following command estimates the same model seen in Figure 11.12, providing further detail.

```
. svy: sem (Enviro -> water1 water2 warming sprawl weather losscen)
      (age sex educ party newcomer -> Enviro), standard
```

(running sem on estimation sample)

Survey: Structural equation model

Number of strata =	1	Number of obs =	734
Number of PSUs =	734	Population size =	725.97798
		Design df =	733

(1) [water1]Enviro = 1

Standardized	Coef.	Linearized Std. Err.	t	P> t	[95% Conf. Interval]
Structural					
Enviro <- age	-.049816	.057598	-0.86	0.387	-.1628926 .0632606
sex	.0613271	.0510951	1.20	0.230	-.0389831 .1616373
educ	.120357	.0491608	2.45	0.015	.0238443 .2168696
party	-.312801	.0500496	-6.25	0.000	-.4110587 -.2145434
newcomer	-.1093371	.0540814	-2.02	0.044	-.2155101 -.0031642
Measurement					
water1 <- Enviro _cons	.6428945 1.310624	.0331035 .1795705	19.42 7.30	0.000 0.000	.5779054 .7078836 .9580899 1.663157
water2 <- Enviro _cons	.6245495 .9240446	.0376486 .1715634	16.59 5.39	0.000 0.000	.5506375 .6984616 .5872303 1.260859
warming <- Enviro _cons	.6783102 1.298958	.0304816 .1864989	22.25 6.96	0.000 0.000	.6184686 .7381518 .9328226 1.665094
sprawl <- Enviro _cons	.6309669 1.125136	.0405904 .1758646	15.54 6.40	0.000 0.000	.5512797 .7106542 .779878 1.470395
weather <- Enviro _cons	.5098348 1.486405	.0405605 .146774	12.57 10.13	0.000 0.000	.4302061 .5894634 1.198257 1.774552
losscen <- Enviro _cons	.6949895 1.181143	.034121 .1923892	20.37 6.14	0.000 0.000	.6280029 .7619762 .8034435 1.558843
Variance					
e.water1	.5866867	.0425642			.5088026 .6764927
e.water2	.6099379	.0470269			.5242621 .7096149
e.warming	.5398953	.0413519			.4645217 .627499
e.sprawl	.6018807	.0512223			.5092724 .7113294
e.weather	.7400685	.0413583			.6631692 .8258849
e.losscen	.5169895	.0474275			.4317826 .619011
e.Enviro	.8653105	.0330935			.8027204 .932781

We find that *educ*, *party* and *newcomer* all have significant effects on the latent variable *Enviro*, whereas *age* and *sex* do not. These results agree with our previous regression of factor score *enviro* on the same five predictors.

The topic of structural equation modeling with Stata deserves at least one book of its own. The examples in this chapter, and those of Chapter 8, take only a first look. Stata's *Structural Equation Modeling Reference Manual* includes a complete description of the commands, along with a library of 26 examples that take the reader much farther.

Time Series Analysis

Stata's time series capabilities are covered in the 700-page *Time-Series Reference Manual*. This chapter provides a brief introduction, beginning with two basic analytical tools: time plots and smoothing. We then move on to illustrate the use of correlograms, ARIMA and ARMAX modeling, together with diagnostic tests for stationarity and white noise. Further applications, notably periodograms and the flexible ARCH family of models, are left to the reader's explorations.

A thorough, technical treatment of time series topics can be found in Hamilton (1994). Other sources include Box, Jenkins and Reinsel (1994), Chatfield (2004), Diggle (1990), Enders (2004) and Shumway (1988).

Menus for time series operations come under the following headings:

Statistics > Time series

Statistics > Multivariate time series

Statistics > Cross-sectional time series

Graphics > Time-series graphs

Example Commands

. ac y, lags(8) level(95) generate(newvar)

Graphs autocorrelations of variable *y*, with 95% confidence intervals (default), for lags 1 through 8. Stores the autocorrelations as the first 8 values of *newvar*.

. arch D.y, arch(1/3) ar(1) ma(1)

Fits an ARCH (autoregressive conditional heteroskedasticity) model for first differences of *y*, including first- through third-order ARCH terms, and first-order AR and MA disturbances.

. arima y, arima(1,1,1)

Fits a simple ARIMA(1,1,1) model, with first differencing and first-order AR and MA terms. Possible options could specify alternative estimation strategies, linear constraints and robust estimates of variance.

- . **arima y, arima(1,0,2) sarima(1,0,1,12)**
Fits ARIMA(1,0,2) \times (1,0,1)₁₂ model with first-order AR terms, first and second-order MA terms, and also a multiplicative seasonal component with period 12.
- . **arima y x1 x2 x3, arima(2,0,1)**
Performs ARMAX (autoregressive moving average with exogenous variables) regression of *y* on three predictors. Errors are modeled as second-order autoregressive and first-order moving average process.
- . **arima D.y x1 L1.x1 x2, ar(1) ma(1 12)**
Fits ARMAX model in which first differences of *y* are regressed on *x1*, lag-1 values of *x1*, and *x2*, including AR(1), MA(1) and MA(12) disturbances.
- . **corrgram y, lags(8)**
Obtains autocorrelations, partial autocorrelations and *Q* tests for lags 1 through 8.
- . **dfuller y**
Performs Dickey–Fuller unit root test for stationarity.
- . **estat dwatson**
After **regress** with time series data, calculates a Durbin–Watson statistic testing first-order autocorrelation.
- . **egen newvar = ma(y), nomiss t(7)**
Generates *newvar* equal to the span-7 moving average of *y*, replacing the start and end values with shorter, uncentered averages.
- . **generate date = mdy(month,day,year)**
Creates variable *date*, equal to days since January 1, 1960, from the three variables *month*, *day*, and *year*.
- . **generate date = date(str_date, "mdy")**
Creates variable *date* from the string variable *str_date*, where *str_date* contains dates in month, day, year form such as “11/19/2001”, “4/18/98”, or “June 12, 1948”. Type **help dates** for many other date functions and options.
- . **generate newvar = L3.y**
Generates *newvar* equal to lag-3 values of *y*.
- . **pac y, lags(8) yline(0) ciopts(bstyle(outline))**
Graphs partial autocorrelations with confidence intervals and residual variance for lags 1 through 8. Draws a horizontal line at 0; shows the confidence interval as an outline, instead of a shaded area (default).
- . **pergram y, generate(newvar)**
Draws the sample periodogram (spectral density function) of variable *y* and creates *newvar* equal to the raw periodogram values.

- **smooth 73 y, generate(newvar)**

Generates *newvar* equal to span-7 running medians of *y*, re-smoothing by span-3 running medians. Compound smoothers such as “3RSSH” or “4253h,twice” are possible. Type **help smooth**, or **help tssmooth**, for other smoothing and filters.

- **tsset date, format(%td)**

Defines the dataset as a time series. Time is indicated by variable *date*, which is formatted as daily. For panel data with parallel time series for a number of different units, such as cities, **tsset city year** identifies both panel and time variables. Most of the commands in this chapter require that the data be **tsset**.

- **tssmooth ma newvar = y, window(2 1 2)**

Applies a moving-average filter to *y*, generating *newvar*. The **window(2 1 2)** option finds a span-5 moving average by including 2 lagged values, the current observation, and 2 leading values in the calculation of each smoothed point. Type **help tssmooth** for a list of other possible filters including weighted moving averages, exponential or double exponential, Holt–Winters and nonlinear.

- **tssmooth nl newvar = y, smoother(4253h,twice)**

Applies a nonlinear smoothing filter to *y*, generating *newvar*. The **smoother(4253h,twice)** option iteratively finds running medians of span 4, 2, 5, and 3, then applies Hanning, and then repeats on the residuals. **tssmooth nl**, unlike other **tssmooth** procedures, cannot work around missing values.

- **wntestq y, lags(15)**

Ljung–Box portmanteau *Q* test for white noise (also provided by **corrgram**).

- **xcorr x y, lags(8) xline(0)**

Graphs cross-correlations between input (*x*) and output (*y*) variable for lags 1–8.

xcorr x y, table gives a text version that includes the actual correlations. If we added a **generate(newvar)** option to this **xcorr** command, it would store correlations for lags from –8 to +8 as a variable in the first 17 rows of the data.

Smoothing

Many time series exhibit high-frequency variations that make it difficult to discern underlying patterns. Smoothing such series breaks the data into two parts, one that varies gradually, and a second “rough” part containing the leftover rapid changes:

$$\text{data} = \text{smooth} + \text{rough}$$

To illustrate smoothing methods, we examine data on daily water consumption for the town of Milford, New Hampshire over seven months from January through July 1983 (*MILwater.dta*; source Hamilton 1985). Milford’s usual patterns of water use were interrupted by alarming news midway through this period.

Contains data from C:\data\MILwater.dta			Milford daily water use, 1/1/83
obs:	212 <th></th> <th>- 7/31/83</th>		- 7/31/83
vars:	4		22 Jun 2012 08:05
size:	1,272		
variable	storage type	display format	value label
month	byte	%9.0g	Month
day	byte	%9.0g	Date
year	int	%9.0g	Year
water	int	%9.0g	Water use in 1000 gallons

Sorted by:

Before further analysis, we need to convert the month, day and year information into a single numerical index of time. Stata's **mdy()** function does this, creating an elapsed-date variable (named *date* here) indicating the number of days since January 1, 1960.

```
. generate date = mdy(month,day,year)
. list in 1/5
```

	month	day	year	water	date
1.	1	1	1983	520	8401
2.	1	2	1983	600	8402
3.	1	3	1983	610	8403
4.	1	4	1983	590	8404
5.	1	5	1983	620	8405

The January 1, 1960 reference date is arbitrary but fixed. We can provide more understandable formatting for *date*, and also set up our data for later analyses, by using the **tsset** (time series set) command to identify *date* as the time index variable and to specify the **%td** (daily) display option for this variable.

```
. tsset date, format(%td)
      time variable: date, 01jan1983 to 31jul1983
                  delta: 1 day
.
. list in 1/5
```

	month	day	year	water	date
1.	1	1	1983	520	01jan1983
2.	1	2	1983	600	02jan1983
3.	1	3	1983	610	03jan1983
4.	1	4	1983	590	04jan1983
5.	1	5	1983	620	05jan1983

Dates in the new *date* format, such as “05jan1983”, are more readable than the underlying numeric values such as “8405” (days since January 1, 1960). If desired, we could use **%td** formatting to produce other formats, such as “05 Jan 1983” or “01/05/83”. Stata offers a number of variable-definition, display-format and dataset-format features that are important with time series. Many of these involve ways to input, convert and display dates. Full descriptions of date functions are found in the *Data Management Reference Manual* and the *User’s Guide*, or they can be explored within Stata by typing **help dates**.

Figure 12.1 uses **twoway line** to draw a simple time plot of *water* against *date*. The graph shows a pattern of day-to-day variation, as well as an upward trend in water use when summer arrives. Values for our date-format variable are labeled automatically (01jan1983, etc.) on the *x* (or *t*) axis, but Stata's default choices here lead to crowded, unsatisfactory results.

```
. graph twoway line water date
```

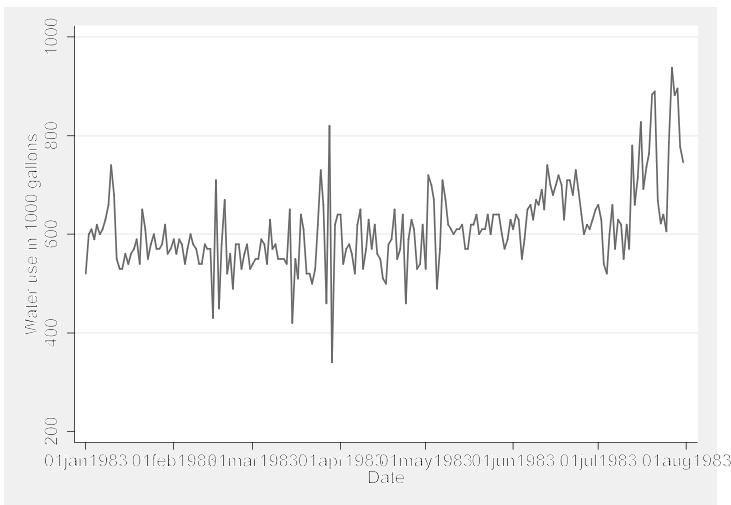


Figure 12.1

A better way to draw time plots when the *x* axis is a date variable is to use the special time series command **tsline**. This command allows us to describe the *x* axis in terms of dates, without having to reference the numerical elapsed dates underneath. For example, we could draw a time plot similar to Figure 12.1 but with a less crowded time axis, as seen in Figure 12.2 on the next page. Note that the **tsline** command does not accept an *x* variable, only one or more *y* variables. With **tsset** data, the time dimension has already been defined. Options **tlabel()** and **ttick()** work in a **tsline** plot just as **xlabel()** and **xtick()** would in any other **twoway** plot, except that they understand date notations such as 01jan1983. In Figure 12.2 we have also suppressed the *x*-axis (time or *t*-axis) title with the option **title("")**, because the word "Date" there seems superfluous below an axis labeled 01jan1983, 01mar1983 and so forth.

```
. tsline water, ylabel(300(100)900) ttitle("")
    tlabel(01jan1983 01mar1983 01may1983 01jul1983, grid)
    ttick(01feb1983 01apr1983 01jun1983 01aug1983)
```

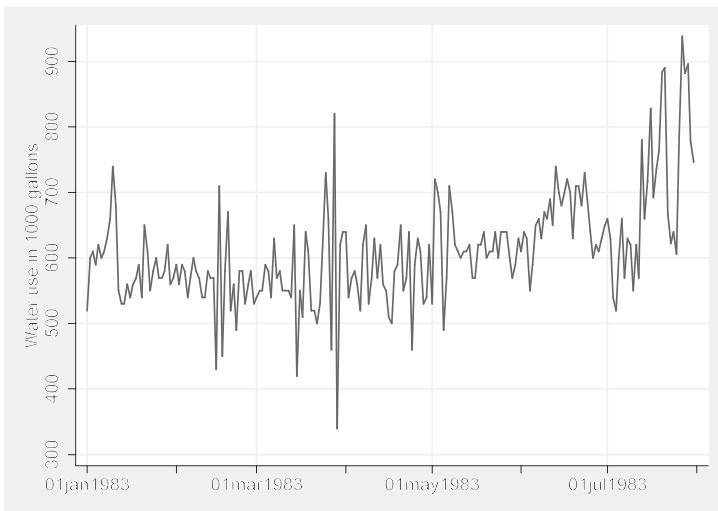


Figure 12.2

Visual inspection plays a key role in time series analysis. Smoothing often helps us to see underlying patterns beneath jagged series. The simplest smoothing method is to calculate a moving average at each data point based on present, earlier and later values of y . For example, a “moving average of span 3” refers to the mean of y_{t-1} , y_t and y_{t+1} . We could use Stata’s explicit subscripting to **generate** such a variable:

```
. generate water3a = (water[_n-1] + water[_n] + water[_n+1])/3
```

A better way involves the **ma** (moving average) function of **egen** :

```
. egen water3b = ma(water), nomiss t(3)
```

The **nomiss** option in this **egen** command requests shorter, uncentered moving averages in the tails; otherwise, the first and last values of $water^3$ would be missing. The **t(3)** option calls for moving averages of span 3. Any odd-number span of 3 or more could be used.

For time series (**tsset**) data, powerful smoothing tools are available through the **tssmooth** commands. All but **tssmooth nl** can handle missing values.

tssmooth ma	moving-average filters, unweighted or weighted
tssmooth exponential	single exponential filters
tssmooth dexpontial	double exponential filters
tssmooth hwinters	nonsseasonal Holt–Winters smoothing
tssmooth shwinters	seasonal Holt–Winters smoothing
tssmooth nl	nonlinear filters

tssmooth ma, for example, can calculate moving averages of span 3, identical to those from our previous **egen** command:

```
. tssmooth ma water3c = water, window(1 1 1)
The smoother applied was
(1/3)*[x(t-1) + 1*x(t) + x(t+1)]; x(t)= water
```

Type **help tssmooth ma**, **help tssmooth exponential**, etc. for the syntax of each command.

Figure 12.3 graphs a 5-day moving average of Milford water use (*water5*), together with the raw data (*water*). The **tsline** command overlays a time plot of smoothed *water5* values onto a plot of raw *water* values (thinner line). *T*-axis labels mark the dates, as they did in Figure 12.2. In Figure 12.3, however, we specify a simpler date display format giving just the month and day, **format(%tdmd)**. The shorter format leaves room to label the beginning of each month in this graph, as opposed to every other month as previously done in Figure 12.2.

```
. tssmooth ma water5 = water, window(2 1 2)
The smoother applied was
(1/5)*[x(t-2) + x(t-1) + 1*x(t) + x(t+1) + x(t+2)]; x(t)= water

. tsline water, clwidth(medium)
    || tsline water5, clwidth(medthick)
    || , ylabel(300(100)900) ytitle("Water use in 1000 gallons")
    tttitle("") tlabel(01jan1983 01feb1983 01mar1983 01apr1983
    01may1983 01jun1983 01jul1983 01aug1983, grid format(%tdmd))
    legend(position(4) ring(0) rows(2)
        label(1 "daily water use") label(2 "5-day average"))
```

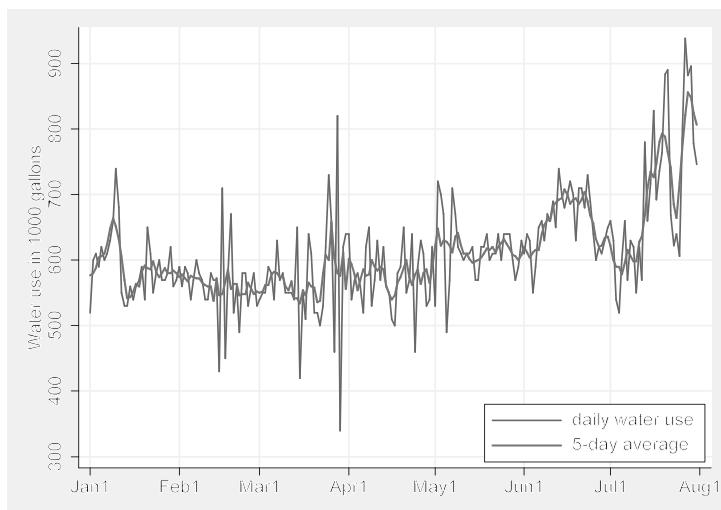


Figure 12.3

Moving averages share a drawback of other mean-based statistics: they have little resistance to outliers. Because outliers form prominent spikes in the water time series, we might try a more resistant smoothing method. The **tssmooth nl** command performs outlier-resistant nonlinear

smoothing, employing methods and terminology described in Velleman and Hoaglin (1981) and Velleman (1982). For example,

```
. tssmooth nl water5r = water, smoother(5)
```

creates a new variable named *water5r*, holding the values of *water* after smoothing by running medians of span 5. Compound smoothers using running medians of different spans, in combination with Hanning ($\frac{1}{4}$, $\frac{1}{2}$ and $\frac{3}{4}$ -weighted moving averages of span 3) and other techniques, can be specified in Velleman's original notation. One compound smoother that seems particularly useful with data that change rapidly is called "4253h, twice." Applying this to *water*, we calculate smoothed variable *water4r*:

```
. tssmooth nl water4r = water, smoother(4253h,twice)
```

Figure 12.4 graphs these new smoothed values, *water4r*. Compare Figure 12.4 with 12.3 to see how 4253h, twice smoothing performs relative to a span-5 moving average. Although both smoothers have similar spans, 4253h, twice does more to reduce the jagged variations.

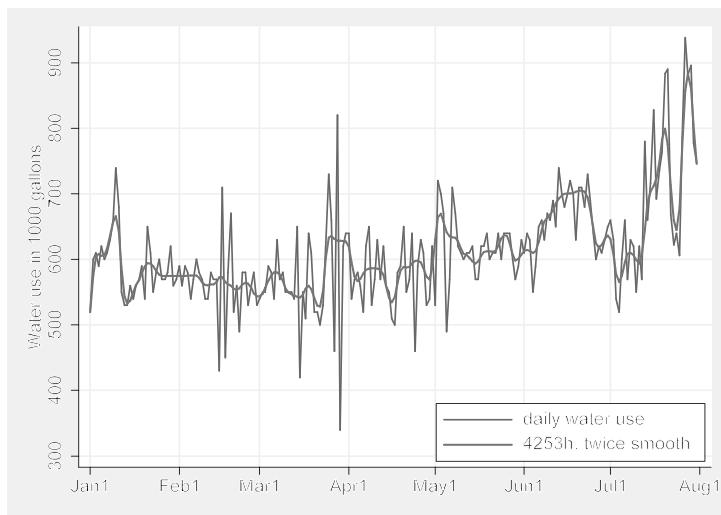


Figure 12.4

Sometimes our goal in smoothing is to look for patterns in smoothed plots. With these particular data, however, the rough or residuals after smoothing actually hold more interest. We can calculate the rough as the difference between data and smooth, and then graph these residuals in their own time plot, as done in Figure 12.5.

```
. generate rough = water - water4r
. label variable rough "Residuals from 4253h, twice"
. tsline rough, ttitle("")
    tlabel(01jan1983 01feb1983 01mar1983 01apr1983
    01may1983 01jun1983 01jul1983 01aug1983, grid format(%tdmd))
```

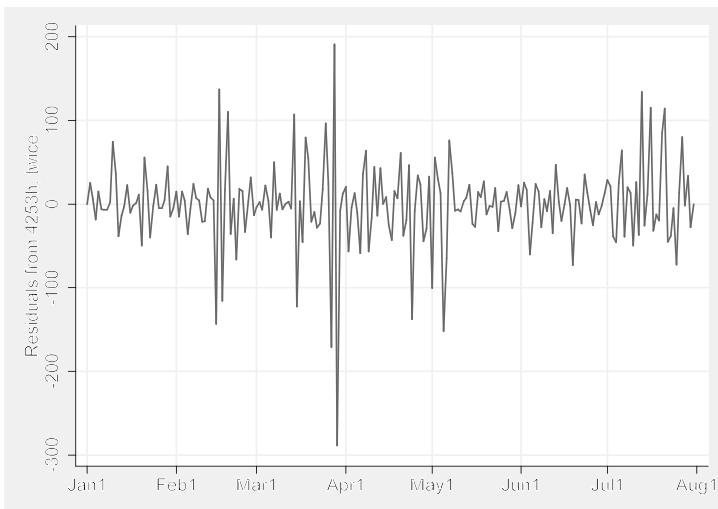


Figure 12.5

The wildest fluctuations in Figure 12.5 occur around March 27–29. Water use abruptly dropped, rose again, and then dropped lower and bounced even higher before settling towards more usual levels. On these days, local newspapers carried stories that hazardous chemical wastes had been discovered in one of the wells that supplied the town’s water. Initial reports alarmed people, and consumption dropped precipitously. Over the next few days, water use vacillated between high and low peaks, in response to new developments or in compensation for delayed use. Things calmed down after the questionable well was taken offline.

Further Time Plot Examples

Dataset *Greenland_temperature.dta* contains a famous time series of temperature estimates reconstructed from the GISP2 ice core in central Greenland, covering time from about 50,000 years ago up through 1855 (Alley 2004). In scientific publications on these data, time has been represented by the variable *age*, in units of thousands of years before present. “Present” had been conventionally defined by the ice researchers to mean 1950. The most recent *age* in their data is .095, or 95 years before present — in other words, 1855. Ice and snow from more recent years had not consolidated enough to apply the temperature reconstruction method. To keep our example more intuitive for non-paleoclimatologists, however, a new time variable, *year*, was generated. *year* can be read as “calendar years” from –48,000 to 1999. The variable *gisptemp* contains temperatures reconstructed from the GISP2 ice core. *shutemp* contains the

average annual temperature at the same Greenland location directly measured by scientists over 1987–1999 (Shuman et al. 2001).

```
. use C:\data\Greenland_temperature.dta, clear
. describe

Contains data from C:\data\Greenland_temperature.dta
obs: 1,646
vars: 4
size: 26,336
      storage   display    value
variable name   type   format   label
year           float  %5.0f   'Calendar' year
age            float  %8.0g   Age, 1,000s of years before 1950
gisptemp       float  %8.0g   GISP2 ice core Central Greenland
                     temp -48,000 to 1855, C
shutemp        float  %9.0g   shuman (2001) summit temp 1987 to
                     1999, C
Sorted by: year
```

Figure 12.6 is a spike plot of the GISP2 temperature data. Spikes are drawn from a baseline equal to the longterm mean, with warmer-than-average temperatures indicated by reddish spikes, and cooler-than-average temperatures by bluish spikes. Because the data are much denser for more recent years, an appropriate baseline for the whole series is calculated from the average of thousand-year periods, rather than the average of individual measurements.

```
. gen millennium = int(year/1000)
. collapse (mean) gisptemp, by(millennium)
. summ gisptemp

      Variable |       Obs        Mean      Std. Dev.       Min       Max
                 | 50  -42.75702  6.798864  -51.48554  -30.02251

. use C:\data\Greenland_temperature.dta, clear
. graph twoway spike gisptemp year if gisptemp > -42.75702,
   base(-42.75702) lcolor(erosc)
|| spike gisptemp year if gisptemp <= -42.75702,
   base(-42.75702) lcolor(eltblue)
|| lowess gisptemp year, bwidth(.05) lwidth(medthick)
   lcolor(black)
|| , ytitle("Central Greenland temperature, `=char(176)'C")
   legend(off)
```

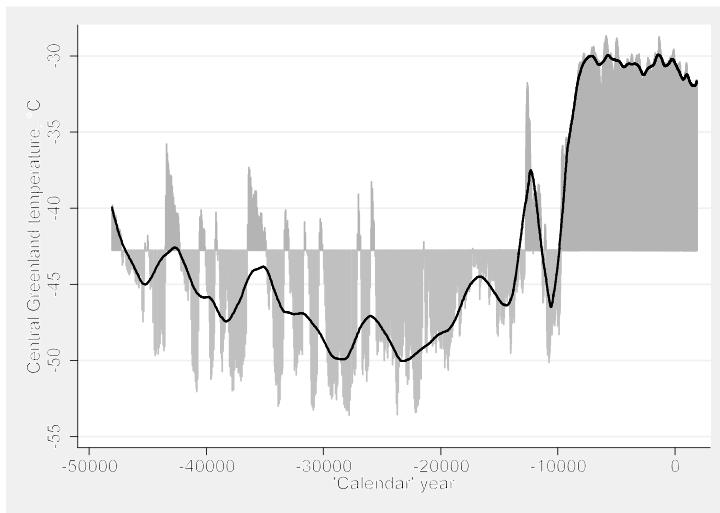


Figure 12.6

This graph clearly shows the transition from Ice Age to warmer conditions. Reconstructed Greenland temperatures rose by about 20°C , from below -50 to around -30°C . The transition was interrupted, however, by a brief return to near-Ice Age conditions called the Younger Dryas, occurring between about 12,900 and 11,500 years before present (calendar years $-10,900$ to $-9,500$ in our graph, the final downward spike). The onset and end of the Younger Dryas each happened over a few decades or less, spurring new research on possible causes and future potential for abrupt climate change (e.g., White et al. 2010).

The moving-average or nonlinear smoothing techniques illustrated in the previous section work best when observations are equally spaced in time, and when much variation occurs over relatively short time spans. The GISP2 ice core measurements have uneven spacing, however, becoming farther apart in time as they descend into deeper (older) and more compressed layers of ice. For time series with uneven spacing, or for smoothing over long spans, lowess regression such as the curve in Figure 12.6 provides a practical alternative.

Figure 12.7 graphs a recent segment of the data in Figure 12.6, depicting GISP2 temperatures for the past 11,000 years only. Graphs much like this particular time plot have been a source of remarkable confusion.

```
. graph twoway line gispstemp year if year > -9000, lwidth(medthick)
    xlabel(-9000(1000)2000, grid gmax gmin)
    title("Greenland ice core only")
    ytitle("Central Greenland temperature, `=char(176) 'C")
```

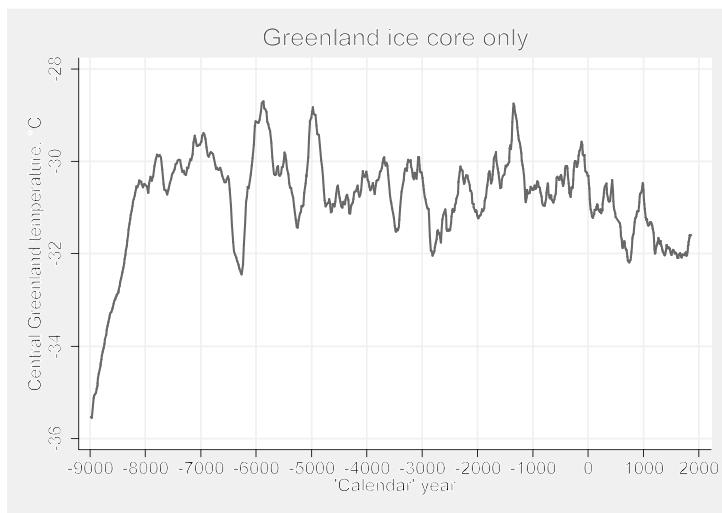


Figure 12.7

One can find many deceptive variations of Figure 12.7 on the Internet, in which the final data point (actually 1855) is implied or labeled to be the “present.” If that were indeed the present temperature, then we might conclude that Greenland remains cooler than average compared with the past 10,000 years, and recent warming has been trivial — which is the point such graphs are drawn to convey. Some are constructed using a baseline plotting method similar to Figure 12.6, but with the baseline defined from that 1855 endpoint to make the “cold now” message even more visually striking. Other variants hide the fact that these are central Greenland temperatures, suggesting instead that they represent the globe.

In 1855, however, Greenland was barely warming from a cold period called the Little Ice Age. Recent temperatures have been warmer, leading to declining ice sheet mass as well as reductions in sea ice around the coast. Even in the 1990s, direct measurements from the summit of the ice sheet reported an average annual temperature of -29.26°C (Shuman et al. 2001). Figure 12.8 re-draws Figure 12.7 but with a final data point showing the 1987–1999 temperature, overlaid as a scatterplot. Two lines of text centered at y coordinate -28.80 , x coordinate 1500 label the scatterplot marker, in the same color as the marker itself.

```
. graph twoway line gisptemp year, lwidth(medthick)
    || scatter shutemp year, msymbol(S) mcolor(red)
    || if year > -9000, xlabel(-9000(1000)2000, grid gmax gmin)
    title("Greenland ice core compared with recent") legend(off)
    ytitle("Central Greenland temperature, `=char(176)'C")
    text(-28.80 1500 "1987`=char(150)`" "1999", color(red))
```

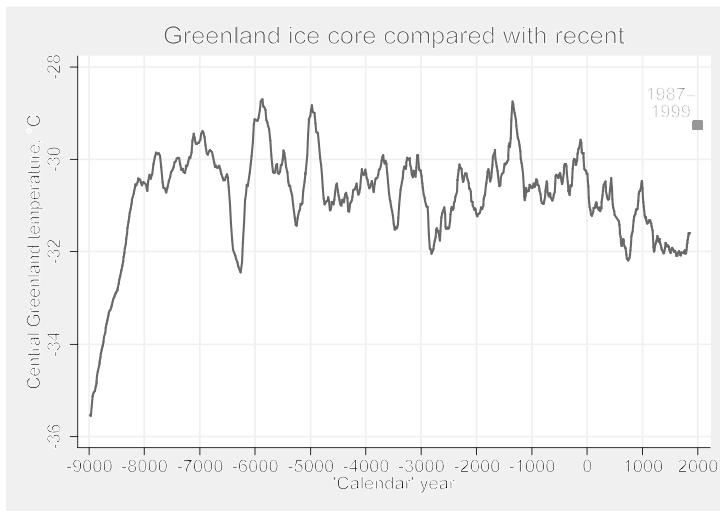
**Figure 12.8**

Figure 12.8 leaves a different impression than Figure 12.7, although the ice-core data in both are the same.

Recent Climate Change

Shifting scale from thousands of years to only the past thirty, the rest of this chapter looks at how climate has changed recently. Dataset *Climate.dta* contains three time series estimating monthly global temperatures from 1980 through 2010, along with four possible drivers or causes of temperature. Two of the temperature indexes derive from surface-temperature measurements (NCDC and NASA), while a third independently estimates lower-troposphere temperatures from satellite data (UAH). See the Dataset Sources appendix for information about all of these variables, brought together from different sources.

Contains data from C:\data\Climate.dta				
obs:	372	Global temperature & drivers 1980-2010		
vars:	11	1 Jul 2012 12:51		
size:	13,764			
variable	storage type	display format	value label	variable label
year	int	%9.0g		Year
month	byte	%9.0g		Month
myear	int	%tmMMCY		Month, year
ncdtemp	float	%9.0g		NCDC global temp anomaly v.1901-2000, C
nasatemp	float	%8.0g		NASA global temp anomaly v.1951-1980, C
uahtemp	float	%9.0g		UAH global temp anomaly v.1981-2010, C

aod	float	%8.0g	Aerosol optical depth at 550nm
tsi1	float	%8.0g	Total Solar Irradiance, W/m ²
mei	float	%9.0g	Multivariate ENSO Index
co2globe	float	%8.0g	Global average marine surface CO ₂ , ppm
co2anom	float	%9.0g	Global CO ₂ anomaly, ppm

Sorted by: myear

The 1980–2010 time window covered by *Climate.dta* was constrained by available data: the global average CO₂ level time series used here begins in 1980, and the Aerosol Optical Depth data had been updated, at this writing, only through 2010. This file is **tsset** by *myear*, a monthly variable defined from separate *month* and *year* variables found in the original data sources. Internally, Stata defines monthly variables as the number of months since January 1960, just as it defines date variables as the number of days since January 1, 1960. Here *myear* has a %tmmCY format, so it will appear to us as Jan1980, Feb1980 and so forth. If not already done, creating this monthly time variable and **tssetting** the data would be necessary steps for later analysis.

```
. gen myear = ym(year, month)
. format myear %tmmCY
. label variable myear "Month, year"
. tsset myear
```

In a widely-cited paper, Foster and Rahmstorf (2011, building on earlier work by Lean and Rind, 2008) analyzed similar temperature and driver variables to uncover “the real global warming signal.” Their analysis is more thorough than the basic examples in this chapter. Nevertheless, we arrive at broadly similar conclusions.

Figure 12.9 plots global temperature anomalies according to the National Climate Data Center (NCDC) index, *ncdctemp*. Temperature anomalies from an individual weather station represent deviations of observed temperature from the long-term mean temperature at that station for that month. Global temperature anomalies are subsequently calculated from a weighted average of many individual-station anomalies around the world. A gray line at zero marks the 20th-century mean.

```
. tsline ncdctemp, lw(medthick)
    ytitle("Global temperature anomaly vs. 1901`=char(150)'2000,
           `=char(176)'C", size(medsmall))
    yline(0, lcolor(gs12) lwidth(thick)) xtitle("")
    xlabel(), grid gmin gmax)
```

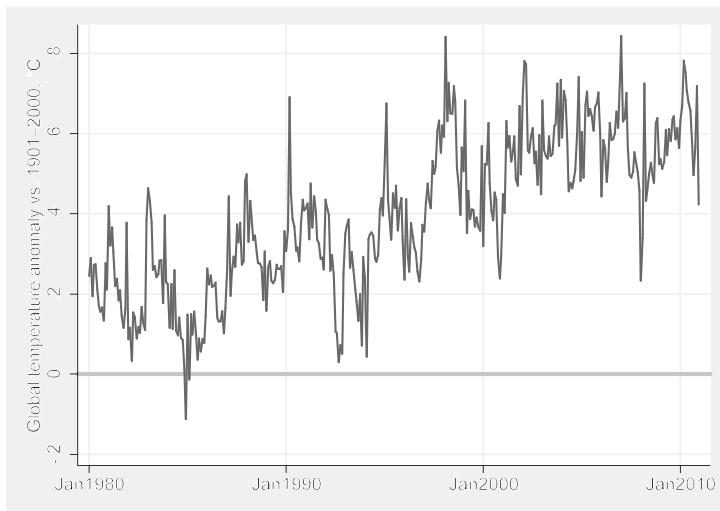


Figure 12.9

All but two months in Figure 12.9 have temperatures above the 20th-century mean. The warming trend has been particularly steep since the mid-1970s, a period mostly captured by this graph. Its central high point is the globally hot month of February 1998, not surpassed (and then just barely) until January 2007. Both are more than two standard deviations higher than any month before 1980 in the NCDC data, which go back to 1880 (see Figure 2.1 in Chapter 2). However, warm temperatures through much of 1998, together with cooler temperatures in some recent years, create a visual impression that global warming might have paused during the past decade. The final section of this chapter applies time series modeling to investigate possible explanations.

If systematic reasons exist, they might be found among the other variables in *Climate.dta*. These include four factors that physical scientists have identified as important drivers or causes of air temperature variations. Opaqueness of the atmosphere, here measured by Aerosol Optical Depth (AOD) at a wavelength of 550 nm, particularly reflects the influence of volcanic eruptions, which cool surface temperatures by injecting sunlight-blocking particles high into the atmosphere (Sato et al. 1993). Total Solar Irradiance (TSI) involves satellite-based measurements of the varying amounts of solar radiation (in Watts per square meter) received at the top of Earth's atmosphere (Fröhlich 2006). El Niño/Southern Oscillation (ENSO) events can substantially affect global air temperatures, either warming air through atmospheric changes that pool unusually warm surface water in the central and eastern tropical Pacific (El Niño), or cooling air temperatures by bringing more deep ocean water to the surface (La Niña). Some ENSO indicators are defined only from sea surface temperatures, creating circularity if then used to “explain” temperature in turn. This Multivariate ENSO Index (MEI), on the other hand, is defined from the first principal component of 6 variables observed over the tropical Pacific: sea-level pressure, zonal and meridional components of the surface wind, sea surface temperature, surface air temperature and total cloudiness fraction of the sky. MEI values substantially above zero indicate El Niño conditions, while values below zero indicate La Niña (Wolter and Timlin 1998). The fourth driver in these data is globally averaged marine-surface

carbon dioxide (CO_2) concentration in parts per million, based on measurements from a network of geographically dispersed sites (Masarie and Tans 1995). Variable *co2globe* gives the actual concentration, and *co2anom* crudely removes seasonal variations by subtracting from each value the 1980–2010 global mean for that month.

Figure 12.10 combines time plots of the four driver series over the same years as Figure 12.9. The AOD plot at top left has a striking appearance, dominated by two great volcanic eruptions: El Chichón in Mexico in late March 1982, and Mount Pinatubo in the Philippines in June 1991. TSI exhibits cyclical behavior, interrupted by sudden high or low spikes and a recent extended quiet period. MEI behavior is not cyclical, although it appears to oscillate with irregular periods between positive and negative conditions. The irregularity and potential for rapid change make El Niño and La Niña events challenging to predict, although their impacts on people make prediction an important goal. CO_2 concentrations have shown far more predictability as they march upward, propelled by roughly 35 gigatons of CO_2 that human activities release to the atmosphere each year.

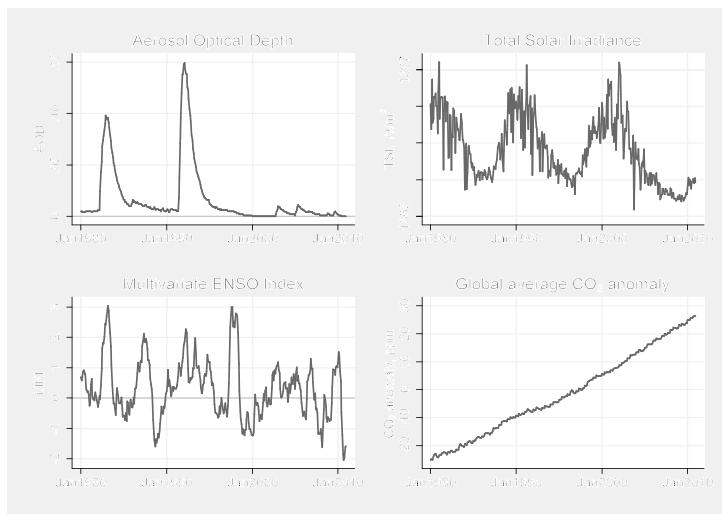


Figure 12.10

The last section of this chapter examines how well these four drivers together explain recent temperature change. Leading up to that analysis, the next sections present basic concepts and tools.

Lags, Leads and Differences

Time series analysis often involves lagged variables, or values from previous times. Lags can be specified by explicit subscripting. For example, the following command creates variable *mei_1*, equal to the previous month's Multivariate ENSO Index (*mei*) value:

```
. generate mei_1 = mei[_n-1]
```

Alternatively, we could accomplish the same thing, using **tsset** data, with Stata's **L.** (lag) operators. Lag operators are simpler than an explicit-subscripting approach. More importantly, the lag operators also respect panel data. The following commands generate 1- and 2-month lagged values of *mei*.

```
. generate mei_1 = L1.mei
. label variable mei_1 "MEI 1-month lag"
. generate mei_2 = L2.mei
. label variable mei_2 "MEI 2-month lag"
. list year month mei mei_1 mei_2 in -5/1
```

	year	month	mei	mei_1	mei_2
368.	2010	8	-1.849	-1.217	-.466
369.	2010	9	-2.037	-1.849	-1.217
370.	2010	10	-1.948	-2.037	-1.849
371.	2010	11	-1.606	-1.948	-2.037
372.	2010	12	-1.58	-1.606	-1.948

We could have obtained this same list without generating any new variables by typing

```
. list year month mei L1.mei L2.mei in -5/1
```

The **L.** operator is one of several that simplify working with time series datasets. Other time series operators are **F.** (lead), **D.** (difference) and **S.** (seasonal difference). These operators can be typed either in upper or lower case — for example, **F2.mei** or **f2.mei**.

Time Series Operators

- L.** Lag y_{t-1} (**L1.** means the same thing)
- L2.** 2-period lag y_{t-2} (similarly, **L3.**, etc. **L(1/4).** means **L1.** through **L4.**)
- F.** Lead y_{t+1} (**F1.** means the same thing)
- F2.** 2-period lead y_{t+2} (similarly, **F3.**, etc.)
- D.** Difference $y_t - y_{t-1}$ (**D1.** means the same thing)
- D2.** Second difference $(y_t - y_{t-1}) - (y_{t-1} - y_{t-2})$ (similarly, **D3.**, etc.)
- S.** Seasonal difference $y_t - y_{t-12}$ (which is the same as **D.**)
- S2.** Second seasonal difference $(y_t - y_{t-12})$ (similarly, **S3.**, etc.)

In the case of seasonal differences, **S12.** does not mean 12th difference, but rather a first difference at lag 12. For example, if we had actual global CO₂ values instead of anomalies, we would see a clear seasonal pattern — lowest in August and September. For some purposes we might want to calculate **S12.co2**, which would be the differences between January 1981 *co2* and January 1980 *co2*, February 1981 *co2* and February 1980 *co2*, and so forth.

Lag operators can appear directly in most analytical commands involving **tsset** data. The following example regresses global temperature (*ncdtemp*) on the previous month's Aerosol Optical Depth (*aod*) index, which in theory should exert a cooling effect. The regression is done without a need to create any new lagged variables.

```
. regress ncdtemp L1.aod
```

Source	SS	df	MS	Number of obs	=	371
Model	1.7221338	1	1.7221338	F(1, 369)	=	54.92
Residual	11.5699674	369	.031354925	Prob > F	=	0.0000
Total	13.2921012	370	.035924598	R-squared	=	0.1296
				Adj R-squared	=	0.1272
				Root MSE	=	.17707

ncdtemp	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
aod _{L1.}	-2.313292	.3121404	-7.41	0.000	-2.92709 -1.699495
_cons	.4361341	.0105842	41.21	0.000	.4153213 .456947

As expected, *aod* has a negative effect on global temperature. The estimated model involves monthly temperature as a function of the previous month's *aod*:

$$\text{predicted } ncdtemp_t = .436 - 2.313aod_{t-1}$$

The coefficient on lagged *aod* (-2.313) appears to be statistically significant ($p \approx 0.000$), but the standard errors and tests in this regression may not be valid. As with any OLS model, they depend on the assumption that errors for successive observations are independent or uncorrelated with each other. Correlated errors occur often in time series analysis, however, so we routinely test for their presence as done in the next section. Standard errors, confidence intervals and tests from OLS regressions involving time series should generally be viewed with suspicion, unless testing shows no evidence of correlated errors.

Although the possibility of correlated errors makes *t* and *F* tests from this regression untrustworthy, the regression equation itself can still provide a valid, least-squares description of the data. Figure 12.11 graphs the predicted values along with observed temperature. The giant Pinatubo eruption in 1991 predicts a substantial cooling, which does appear in the temperature data. Obviously, however, there is much more besides these two volcanoes affecting global climate.

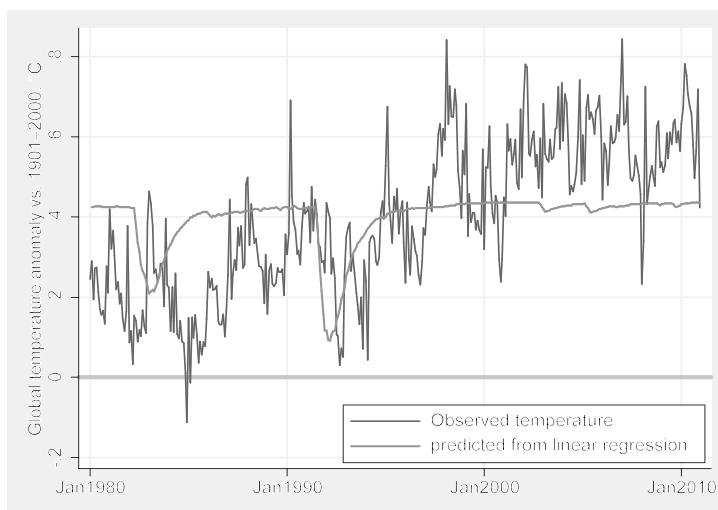


Figure 12.11

Continuing for the moment in a purely descriptive mode, we could explore whether other proposed temperature drivers in these data improve the fit to observed temperatures. Including lagged solar irradiance as a second predictor raises R^2_a slightly, from .127 to .144. The coefficient on lagged solar irradiance is negative, however, which makes no physical sense.

. regress ncdctemp L1.aod L1.tsil

Source	SS	df	MS	Number of obs = 371 F(2, 368) = 32.22 Prob > F = 0.0000 R-squared = 0.1490 Adj R-squared = 0.1444 Root MSE = .17532		
Model	1.98067563	2	.990337816			
Residual	11.3114256	368	.03073757			
Total	13.2921012	370	.035924598			
ncdctemp	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
aod L1.	-2.172641	.3128342	-6.95	0.000	-2.787808	-1.557474
tsil L1.	-.0598584	.0206393	-2.90	0.004	-.1004441	-.0192727
_cons	82.19402	28.19026	2.92	0.004	26.75982	137.6282

Adding a third predictor, lagged Multivariate ENSO Index, raises the explained variance further, $R^2_a = .201$. The coefficient on L1.mei is positive, as it should be given the known warming effect of El Niño. We still see an implausible negative coefficient on L1.tsil, however.

. regress ncdctemp L1.aod L1.tsil L1.mei

Source	SS	df	MS	Number of obs = 371 F(3, 367) = 32.00 Prob > F = 0.0000 R-squared = 0.2074 Adj R-squared = 0.2009 Root MSE = .16943		
Model	2.75629902	3	.918766339			
Residual	10.5358022	367	.028707908			
Total	13.2921012	370	.035924598			
ncdctemp	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
aod L1.	-2.949131	.3372229	-8.75	0.000	-3.612262	-2.285999
tsil L1.	-.05509	.0199673	-2.76	0.006	-.0943546	-.0158253
mei L1.	.0524371	.0100882	5.20	0.000	.0325991	.072275
_cons	75.67739	27.27247	2.77	0.006	22.04748	129.3073

The most substantial improvement in R^2_a occurs when we include CO₂ anomaly among the predictors. Together, these four drivers now explain 72.7% of the variance in monthly temperatures. CO₂ anomaly has by far the strongest effect, in a positive direction as expected from the physics of greenhouse gases. Once we control for CO₂, the coefficient on solar irradiance becomes positive as well.

. regress ncdctemp L1.aod L1.tsil L1.mei L1.co2anom

Source	SS	df	MS	Number of obs = 371 F(4, 366) = 247.57 Prob > F = 0.0000 R-squared = 0.7301 Adj R-squared = 0.7272 Root MSE = .099			
Model	9.70518563	4	2.42629641				
Residual	3.58691559	366	.009800316				
Total	13.2921012	370	.035924598				

ncdctemp	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
aod L1.	-1.535808	.2040555	-7.53	0.000	-1.937077 -1.13454
tsi1 L1.	.0882862	.012849	6.87	0.000	.0630189 .1135534
mei L1.	.0689124	.0059267	11.63	0.000	.0572578 .0805671
co2anom L1.	.0109831	.0004125	26.63	0.000	.010172 .0117942
_cons	-120.1742	17.55028	-6.85	0.000	-154.6862 -85.66217

Figure 12.12 plots the progressive improvement of these four models. Predicted values in the lower right plot, based on the regression with all four drivers, track observed temperature remarkably well — including details such as the cooling after Pinatubo erupted, a warm spike during the “super El Niño” of 1998, and the wandering path of the past decade, cooled by low solar irradiance and low to negative ENSO. The fit between observed and predicted temperatures is striking also because the predictions come from an extremely simplified model in which all effects are linear, and operate with just one-month lags.

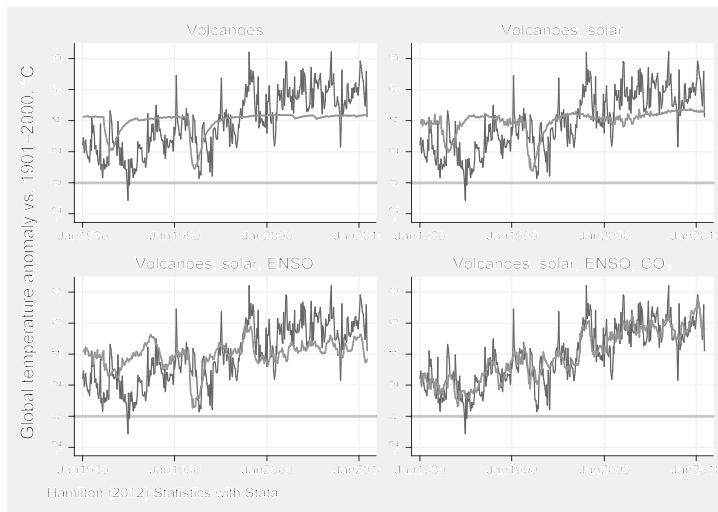


Figure 12.12

Global climate models based on physics instead of statistics, incorporating many more variables and vastly greater geographic complexity, may take weeks to run on a supercomputer. The appeal of the results graphed in Figure 12.12 is that such a simplified model performs as well

as it does, given its limitations. One of these limitations is statistical: OLS standard errors are biased, and the usual t and F tests invalid, if correlations occur among the errors. One simple check for correlated errors, called the Durbin–Watson test, can be run after any regression.

```
. estat dwatson
Durbin-Watson d-statistic( 5, 371) = 1.131453
```

Statistical textbooks provide tables of critical values for the Durbin–Watson test. Given 5 estimated parameters (4 predictors and the y intercept) and 371 observations, these critical values are approximately $d_L = 1.59$ and $d_U = 1.76$. A test statistic below $d_L = 1.59$ leads to rejecting the null hypothesis of no positive first-order (lag 1) autocorrelation. Because our calculated statistic, 1.131, is well below $d_L = 1.59$, we should reject this null hypothesis and conclude instead that there does exist positive first-order autocorrelation. This finding confirms earlier suspicions about the validity of tests in our OLS models for temperature.

Had the calculated statistic been above $d_U = 1.76$, we would fail to reject the null hypothesis. That is, we would then have no evidence of significant autocorrelation. Calculated Durbin–Watson statistics between d_L and d_U are inconclusive, neither rejecting nor failing to reject H_0 .

The Durbin–Watson statistic tests first-order autocorrelation, and ordinarily considers only the alternative hypothesis that it is positive. In practice, autocorrelation can be negative or positive, and can occur at other lags besides 1. The next section presents more general diagnostic tools.

Correlograms

Autocorrelation coefficients estimate the correlation between a variable and itself at particular lags. For example, first-order autocorrelation is the correlation between y_t and y_{t-1} . Second order refers to $\text{Cor}[y_t, y_{t-2}]$, and so forth. A correlogram graphs correlation versus lag.

Stata's **corrgram** command provides simple correlograms and related information. The maximum number of lags it shows can be limited by the data, by **matsize**, or to some arbitrary lower number that is set by specifying the **lags()** option:

```
. corrgram mei, lags(13)
```

LAG	AC	PAC	Q	Prob>Q	-1 [Autocorrelation]	0 [Partial Autocor]	1
1	0.9473	0.9569	336.5	0.0000			
2	0.8582	-0.4181	613.45	0.0000			
3	0.7532	-0.0631	827.33	0.0000			
4	0.6350	-0.1578	979.77	0.0000			
5	0.5167	0.0033	1081	0.0000			
6	0.4036	-0.0680	1142.9	0.0000			
7	0.2983	-0.0299	1176.8	0.0000			
8	0.2060	-0.0235	1193	0.0000			
9	0.1224	-0.0393	1198.8	0.0000			
10	0.0499	-0.0185	1199.7	0.0000			
11	-0.0140	-0.0359	1199.8	0.0000			

12	-0.0723	-0.0340	1201.8	0.0000		
13	-0.1243	-0.0456	1207.8	0.0000		

Lags appear at the left side of the table, followed by columns of autocorrelations (AC) and partial autocorrelations (PAC). For example, the correlation between mei_t and mei_{t-2} is .8582, and the partial autocorrelation (adjusted for lag 1) is -.4181. The Q statistics (Ljung–Box portmanteau) test a series of null hypotheses that all autocorrelations up to and including a given lag are zero. Because the p -values seen here are all very small, we should reject the null hypotheses and conclude that the Multivariate ENSO Index (mei) exhibits significant autocorrelation. If none of the Q statistic probabilities had been below .05, we might conclude instead that the series was white noise with no significant autocorrelation.

At right in a **corrgram** output table are character-based plots of autocorrelations and partial autocorrelations. Inspection of such plots plays a role in choosing time series models. Graphical autocorrelation plots can be obtained through the **ac** command:

```
. ac mei, lags(25)
```

The resulting correlogram, Figure 12.13, includes a shaded area marking pointwise 95% confidence intervals. Correlations outside of these intervals are individually significant. The **lags(25)** option extends this graph to a 25-month lag, revealing that autocorrelations of mei become negative after about 12 months. This pattern reflects the quasi-oscillatory character of ENSO: high periods have some tendency to be followed by low ones, and vice versa.

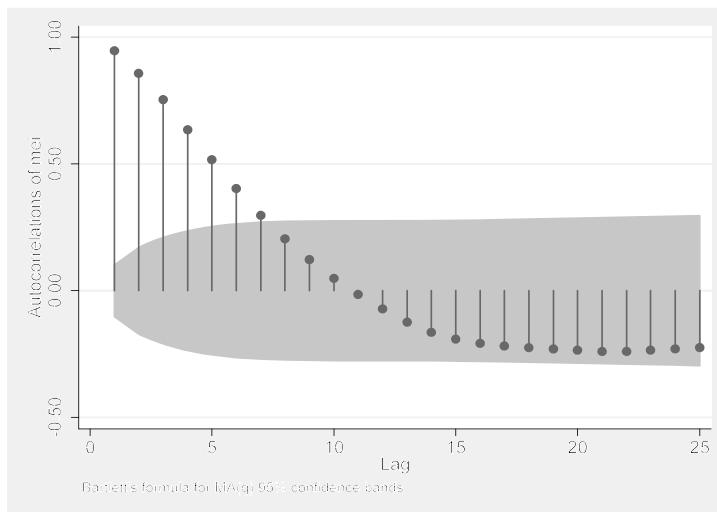
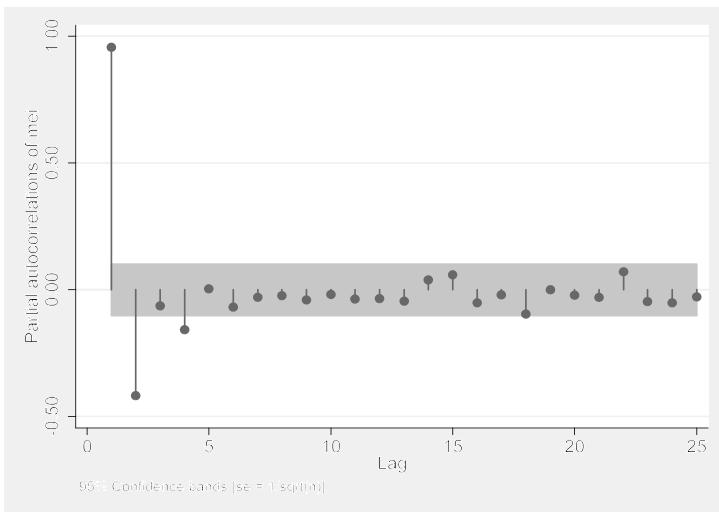


Figure 12.13

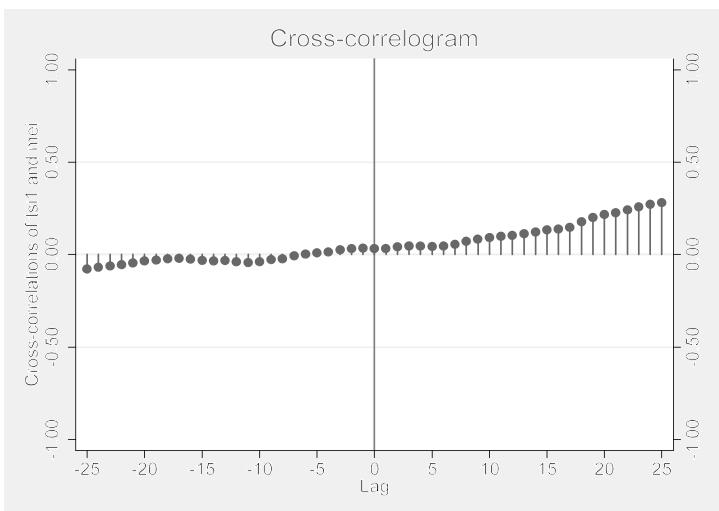
A similar command, **pac**, produces the graph of partial autocorrelations in Figure 12.14. Approximate confidence intervals are based on standard error estimates of $1/\sqrt{n}$. The partial autocorrelations of mei , unlike the autocorrelations, cut off to mostly nonsignificant values after a 2-month lag.

```
. pac mei, lags(25)
```

**Figure 12.14**

Cross-correograms help to explore relationships between two time series. Figure 12.15 shows the cross-correogram of Total Solar Irradiance (*tsi1*) and *mei*. The cross-correlations are near zero for negative lags, and gradually become stronger (though still weak) at longer positive lags. There is virtually no correlation within the same year.

```
. xcorr tsi1 mei, lags(25) xline(0) xlabel(-25(5)25)
```

**Figure 12.15**

Cross-correlograms are easiest to read if we list our input or independent variable first in the **xcorr** command, and the output or dependent variable second—as done for Figure 12.15. Then positive lags denote correlations between input at time t and output at time $t+1, t+2$ etc. In this graph we see some correlation between solar irradiance and the state of ENSO about two years later, consistent with a mild, delayed solar influence on ENSO. There is no correlation between solar irradiance and the state of ENSO two years earlier, giving (sensibly) no reason to think that ENSO affects the sun.

The actual cross-correlation coefficients, and a text version of the cross-correlogram, can be obtained with the **table** option:

```
. xcorr tsil mei, lags(13) table
```

LAG	CORR	-1	0	1
		[Cross-correlation]		
-13	-0.0308			
-12	-0.0370			
-11	-0.0434			
-10	-0.0372			
-9	-0.0275			
-8	-0.0209			
-7	-0.0062			
-6	0.0038			
-5	0.0093			
-4	0.0156			
-3	0.0262			
-2	0.0343			
-1	0.0355			
0	0.0326			
1	0.0340			
2	0.0427			
3	0.0466			
4	0.0468			
5	0.0441			
6	0.0476			
7	0.0558			
8	0.0721			
9	0.0849			
10	0.0930			
11	0.1008			
12	0.1041			
13	0.1142			

The remainder of this chapter goes beyond correlation to more formal time series modeling.

ARIMA Models

Autoregressive integrated moving average (ARIMA) models can be estimated through the **arima** command. This command encompasses autoregressive (AR), moving average (MA), or ARIMA models. It also can estimate structural models that include one or more predictor variables and ARIMA disturbances. These are termed ARMAX models, for autoregressive moving average with exogenous variables. The general form of such ARMAX models, in matrix notation, is

$$y_t = \mathbf{x}_t \boldsymbol{\beta} + \mu_t \quad [12.1]$$

where y_t is the vector of dependent-variable values at time t , \mathbf{x}_t is a matrix of exogenous predictor-variable values (usually including a constant), and μ_t is a vector of “everything else” disturbances. Those disturbances can be autoregressive or moving-average, of any order. For example, ARMA(1,1) disturbances are

$$\mu_t = \rho\mu_{t-1} + \theta\epsilon_{t-1} + \epsilon_t \quad [12.2]$$

where ρ is a first-order autoregressive parameter, θ is a first-order moving average parameter, and ϵ_t represents random, uncorrelated white-noise (normal i.i.d.) errors. **arima** fits simple models as a special case of [12.1] and [12.2], with a constant (β_0) replacing the structural term $\mathbf{x}_t\beta$. Therefore, a simple ARMA(1,1) model becomes

$$\begin{aligned} y_t &= \beta_0 + \mu_t \\ &= \beta_0 + \rho\mu_{t-1} + \theta\epsilon_{t-1} + \epsilon_t \end{aligned} \quad [12.3]$$

Some sources present an alternative version. In the ARMA(1,1) case, they show y_t as a function of the previous y value (y_{t-1}) and the present (ϵ_t) and lagged (ϵ_{t-1}) disturbances:

$$y_t = \alpha + \rho y_{t-1} + \theta\epsilon_{t-1} + \epsilon_t \quad [12.4]$$

Because in the simple structural model $y_t = \beta_0 + \mu_t$, equation [12.3] (employed by Stata) is equivalent to [12.4], apart from rescaling the constant $\alpha = (1-\rho)\beta_0$.

Using the **arima** command, an ARMA(1,1) model can be specified in either of two ways:

```
. arima y, ar(1) ma(1)
```

or

```
. arima y, arima(1,0,1)
```

The **i** in **arima** stands for “integrated,” referring to models that also involve differencing. To fit an ARIMA(2,1,1) model, use

```
. arima y, arima(2,1,1)
```

or equivalently,

```
. arima D.y, ar(1/2) ma(1)
```

Note that within the **arima()** option, the number given for autoregressive or moving average terms specifies all lags up to and including that number, so “2” means both lags 1 and 2. When the **ar()** or **ma()** options are used, however, the number specifies only that particular lag, so “2” means a lag of 2 only. Either command above specifies a model in which first differences of the dependent variable ($y_t - y_{t-1}$) are a function of first differences one and two lags previous ($y_{t-1} - y_{t-2}$ and $y_{t-2} - y_{t-3}$), and also of present and previous errors (ϵ_t and ϵ_{t-1}).

To estimate a structural model in which y_t depends on two predictor variables x (present and lagged values, x_t and x_{t-1}) and w (present values only, w_t), with ARIMA(1,0,3) errors and also multiplicative seasonal ARIMA(1,0,1)₁₂ errors looking back 12 periods (as might be appropriate for monthly data, over a number of years), a suitable command could have the form

```
. arima y x L.x w, arima(1,0,3) sarima(1,0,1,12)
```

In econometric notation, this corresponds to an ARIMA(1,0,3)×(1,0,1)₁₂ model.

A time series y is considered stationary if its mean and variance do not change with time, and if the covariance between y_t and y_{t+u} depends only on the lag u , and not on the particular values of t . ARIMA modeling assumes that our series is stationary, or can be made stationary through appropriate differencing or transformation. We can check this assumption informally by inspecting time plots for trends in level or variance. We have already seen that global temperature, for example, exhibits a clear trend, suggesting that it is not stationary.

Formal statistical tests for “unit roots,” a nonstationary AR(1) process in which $\rho_1 = 1$ (also known as a random walk), supplement theory and informal inspection. Stata offers three unit root tests: **pperron** (Phillips–Perron), **dfuller** (augmented Dickey–Fuller) and **dfgls** (augmented Dickey–Fuller using GLS). **dfgls** is the most powerful and informative.

The table below applies **dfgls** to the National Climate Data Center (NCDC) global temperature anomalies.

```
. dfgls ncdctemp, notrend
```

DF-GLS for ncdctemp

Maxlag = 16 chosen by Schwert criterion

Number of obs = 355

[lags]	DF-GLS mu Test Statistic	1% Critical Value	5% Critical Value	10% Critical Value
16	-1.162	-2.580	-1.952	-1.637
15	-1.191	-2.580	-1.955	-1.640
14	-1.179	-2.580	-1.958	-1.643
13	-1.170	-2.580	-1.962	-1.646
12	-1.194	-2.580	-1.965	-1.649
11	-1.264	-2.580	-1.968	-1.652
10	-1.190	-2.580	-1.971	-1.655
9	-1.204	-2.580	-1.974	-1.658
8	-1.525	-2.580	-1.977	-1.660
7	-1.462	-2.580	-1.980	-1.663
6	-1.639	-2.580	-1.982	-1.665
5	-1.844	-2.580	-1.985	-1.668
4	-1.968	-2.580	-1.988	-1.670
3	-2.058	-2.580	-1.990	-1.672
2	-2.342	-2.580	-1.993	-1.675
1	-2.731	-2.580	-1.995	-1.677

Opt Lag (Ng-Perron seq t) = 9 with RMSE .0915337

Min SC = -4.687887 at lag 1 with RMSE .0943745

Min MAIC = -4.721196 at lag 9 with RMSE .0915337

The **dfgls** output above reports tests of the nonstationary null hypothesis — that the temperature series represents a random walk, or has a unit root — for lags from 1 to 16 months. At the bottom, the output offers three different methods for choosing an appropriate number of lags: Ng–Perron sequential t , minimum Schwarz information criteria, and Ng–Perron modified

Akaike information criteria (MAIC). The MAIC is more recently developed, and Monte Carlo experiments support its advantages over the Schwarz method. Both sequential t and MAIC recommend 9 lags. The DF-GLS statistic for 9 lags is -1.204 , greater than the 10% critical value of -1.658 , so we should not reject the null hypothesis. These results confirm earlier impressions that the *ncdctemp* time series is not stationary.

A similar test of the Multivariate ENSO Index, on the other hand, rejects the nonstationary null hypothesis at all lags, even at the 1% level.

```
. dfgls mei, notrend
```

		Number of obs = 355		
[lags]	DF-GLS mu Test Statistic	1% Critical Value	5% Critical Value	10% Critical Value
16	-3.686	-2.580	-1.952	-1.637
15	-3.742	-2.580	-1.955	-1.640
14	-3.681	-2.580	-1.958	-1.643
13	-4.062	-2.580	-1.962	-1.646
12	-4.381	-2.580	-1.965	-1.649
11	-4.352	-2.580	-1.968	-1.652
10	-4.420	-2.580	-1.971	-1.655
9	-4.451	-2.580	-1.974	-1.658
8	-4.589	-2.580	-1.977	-1.660
7	-4.604	-2.580	-1.980	-1.663
6	-4.655	-2.580	-1.982	-1.665
5	-4.699	-2.580	-1.985	-1.668
4	-4.591	-2.580	-1.988	-1.670
3	-4.909	-2.580	-1.990	-1.672
2	-4.293	-2.580	-1.993	-1.675
1	-4.049	-2.580	-1.995	-1.677
Opt Lag (Ng-Perron seq t) = 3 with RMSE .2688427				
Min SC = -2.565539 at lag 1 with RMSE .2727197				
Min MAIC = -2.497381 at lag 1 with RMSE .2727197				

For a stationary series such as *mei*, correlograms provide guidance about selecting a preliminary ARIMA model:

- AR(p) An autoregressive process of order p has autocorrelations that damp out gradually with increasing lag. Partial autocorrelations cut off after lag p .
- MA(q) A moving average process of order q has autocorrelations that cut off after lag q . Partial autocorrelations damp out gradually with increasing lag.
- ARMA(p,q) A mixed autoregressive-moving average process has autocorrelations and partial autocorrelations that damp out gradually with increasing lag.

Correlogram spikes at seasonal lags (for example, at 12, 24 or 36 in monthly data) indicate a seasonal pattern. Identification of seasonal models follows similar guidelines, applied to autocorrelations and partial autocorrelations at seasonal lags.

We have seen that autocorrelations for *mei* damp out gradually with increasing lag (Figure 12.14), while partial autocorrelations cut off after lag 2. These correlogram patterns, together with **dfgl** test results supporting stationarity, suggest that **mei** could be modeled as an ARIMA(2,0,0) process.

```
. arima mei, arima(2,0,0) nolog
ARIMA regression
Sample: 1 - 372
Number of obs = 372
Wald chi2(2) = 4385.66
Prob > chi2 = 0.0000
Log Likelihood = -43.61494



|      | mei    | Coef.     | OPG Std. Err. | z      | P> z  | [95% Conf. Interval] |
|------|--------|-----------|---------------|--------|-------|----------------------|
| mei  | _cons  | .2814275  | .2132974      | 1.32   | 0.187 | -.1366278 .6994828   |
| ARMA | ar     |           |               |        |       |                      |
|      | L1.    | 1.349918  | .0424774      | 31.78  | 0.000 | 1.266664 1.433173    |
|      | L2.    | -.4163392 | .0415425      | -10.02 | 0.000 | -.497761 -.3349174   |
|      | /sigma | .2710638  | .0091183      | 29.73  | 0.000 | .2531922 .2889354    |


```

Note: The test of the variance against zero is one sided, and the two-sided confidence interval is truncated at zero.

This ARIMA(2,0,0) model represents *mei* as a function of disturbances (μ) from one and two previous months, plus random white-noise errors (ϵ):

$$y_t = \beta_0 + \rho_1 \mu_{t-1} + \rho_2 \mu_{t-2} + \epsilon_t \quad [12.5]$$

where y_t is the value of *mei* at time t . The output table gives parameter estimates $\beta_0 = .28$, $\rho_1 = 1.35$ and $\rho_2 = -.42$.

After we fit an **arima** model, these estimates and other results are saved temporarily in Stata's usual way. For example, to see the model's AR(1) coefficient and standard error, type

```
. display [ARMA]_b[L1.ar]
1.3499184

. display [ARMA]_se[L1.ar]
.04247738
```

Both the first and second-order autoregressive coefficients in this example are significant and far from zero ($t = 31.78$ and -10.02 , respectively), which gives one indication of model adequacy. We can obtain predicted values, residuals and other case statistics after **arima** through **predict**:

```
. predict meihat
. label variable meihat "predicted MEI"
. predict meires, resid
. label variable meires "residual MEI"
```

Graphically, predicted values from this model appear almost indistinguishable from the observed temperatures (Figure 12.16). This image highlights how closely ARIMA models can fit strongly autocorrelated series, by predicting a variable from its own previous values, plus the values of previous disturbances. The ARIMA(2,0,0) model explains about 92% of the variance in *mei*.

```
. tsline mei meihat, lcolor(blue red) lwidth(medium medthick)
    xtitle("") xlabel(, grid gmax gmin) ytitle("MEI")
    ylabel(-2(1)3, grid gmin gmax) yline(0, lcolor(gs12))
```

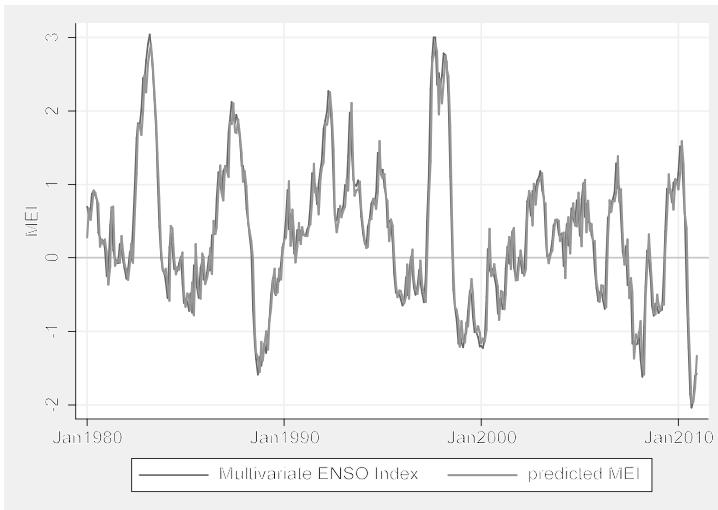


Figure 12.16

An important test of ARIMA model adequacy is whether the residuals appear to be uncorrelated white noise. **corrgram** tests the white-noise null hypothesis across different lags.

```
. corrgram meires, lags(13)
```

LAG	AC	PAC	Q	Prob>Q	-1 [Autocorrelation]	0 [Partial Autocor]	1 -1	0 [Partial Autocor]	1 [Autocor]
1	-0.0241	-0.0241	.2176	0.6409					
2	-0.0099	-0.0105	.25451	0.8805					
3	0.1430	0.1430	7.9641	0.0468					
4	-0.0015	0.0049	7.965	0.0929					
5	0.0256	0.0292	8.2135	0.1449					
6	0.0200	0.0010	8.3656	0.2125					
7	-0.0305	-0.0315	8.7197	0.2734					
8	-0.0001	-0.0109	8.7197	0.3665					
9	-0.0306	-0.0367	9.0787	0.4300					
10	-0.0274	-0.0256	9.3682	0.4976					
11	-0.0292	-0.0331	9.6962	0.5579					
12	-0.0009	0.0079	9.6966	0.6426					
13	-0.0449	-0.0404	10.477	0.6545					

corrgram's portmanteau *Q* tests find no significant autocorrelation among residuals out to lag 13, apart from one just below the .05 level at lag 3. We could obtain the same test for any specific lag using a **wntestq** command, such as

```
. wntestq meires, lags(13)
Portmanteau test for white noise
Portmanteau (Q) statistic = 10.4772
Prob > chi2(13) = 0.6545
```

Thus, the ARIMA(2,0,0) model originally proposed after looking at correlogram patterns turns out to fit quite well, with both of its AR terms significant. It leaves residuals that pass tests for white noise.

Our earlier **dfgls, notrend** test for global temperature anomalies (*ncdctemp*), unlike the test for Multivariate ENSO Index (*mei*), found that temperature appears nonstationary because it contains a prominent trend. A default **dfgls** test that includes a linear trend rejects the nonstationary null hypothesis at all lags.

```
. dfgls ncdctemp
DF-GLS for ncdctemp
Maxlag = 16 chosen by Schwert criterion
Number of obs = 355
[lags] DF-GLS tau Test Statistic 1% Critical Value 5% Critical Value 10% Critical Value
16 -3.624 -3.480 -2.818 -2.536
15 -3.625 -3.480 -2.824 -2.542
14 -3.552 -3.480 -2.829 -2.547
13 -3.489 -3.480 -2.835 -2.552
12 -3.486 -3.480 -2.840 -2.557
11 -3.566 -3.480 -2.846 -2.562
10 -3.395 -3.480 -2.851 -2.566
9 -3.377 -3.480 -2.856 -2.571
8 -3.861 -3.480 -2.861 -2.575
7 -3.697 -3.480 -2.865 -2.580
6 -3.950 -3.480 -2.870 -2.584
5 -4.251 -3.480 -2.874 -2.588
4 -4.399 -3.480 -2.879 -2.592
3 -4.479 -3.480 -2.883 -2.595
2 -4.920 -3.480 -2.887 -2.599
1 -5.505 -3.480 -2.891 -2.602
Opt Lag (Ng-Perron seq t) = 9 with RMSE .0902188
Min SC = -4.749446 at lag 1 with RMSE .0915139
Min MAIC = -4.639686 at lag 9 with RMSE .0902188
```

First differencing time series will remove a linear trend. First differences of temperature show a pattern of autocorrelations that cut off after lag 1, and partial autocorrelations that damp out after lag 1.

```
. corrgram D.ncdctemp, lag(13)
LAG AC PAC Q Prob>Q -1 0 1 -1 0 1
1 -0.3941 -0.4031 58.094 0.0000
2 0.0425 -0.1355 58.773 0.0000
3 -0.0639 -0.1174 60.307 0.0000
4 0.0305 -0.0479 60.656 0.0000
5 -0.0359 -0.0614 61.145 0.0000
6 -0.0385 -0.1032 61.706 0.0000
7 -0.0035 -0.0872 61.711 0.0000
8 0.0633 0.0164 63.239 0.0000
9 -0.1316 -0.1432 69.861 0.0000
10 0.1031 -0.0149 73.938 0.0000
```

11	0.0030	0.0359	73.942	0.0000
12	-0.0353	-0.0393	74.421	0.0000
13	0.0099	-0.0175	74.459	0.0000

These observations suggest that an ARIMA(0,1,1) model might be appropriate for temperatures.

```
. arima ncdctemp, arima(0,1,1) nolog
```

ARIMA regression

Sample: 2 - 372	Number of obs = 371
Log Likelihood = 355.3385	Wald chi2(1) = 152.87
	Prob > chi2 = 0.0000

D.ncdctemp	Coef.	OPG Std. Err.	z	P> z	[95% Conf. Interval]	
ncdctemp _cons	.0007623	.0025694	0.30	0.767	-.0042737	.0057982
ARMA						
ma L1.	-.4994929	.040399	-12.36	0.000	-.5786735	-.4203124
/sigma	.0928235	.0029514	31.45	0.000	.0870388	.0986081

Note: The test of the variance against zero is one sided, and the two-sided confidence interval is truncated at zero.

This ARIMA(0,1,1) model describes the first difference or month-to-month change in temperature as a function of present and 1-month lagged random noise:

$$y_t - y_{t-1} = \beta_0 + \theta \epsilon_{t-1} + \epsilon_t \quad [12.6]$$

where y_t represents *ncdctemp* at time t . Parameter estimates are $\beta_0 = .00076$ and $\theta = -.499$. The first-order MA term, θ , is statistically significant ($p \approx 0.000$), and the model's residuals are indistinguishable from white noise.

```
. predict Dncdchat
. label variable Dncdchat "predicted 1st diff temp"
. predict Dncdcres, resid
. label variable Dncdcres "residual 1st diff temp"
. corrgram Dncdcres, lag(13)
```

LAG	AC	PAC	Q	Prob>Q	-1	0	[Autocorrelation]	1	-1	0	1	[Partial Autocor]
1	0.0133	0.0135	.06574	0.7976								
2	0.0262	0.0265	.32335	0.8507								
3	-0.0631	-0.0657	1.8204	0.6105								
4	-0.0313	-0.0306	2.1907	0.7007								
5	-0.0848	-0.0827	4.912	0.4267								
6	-0.0867	-0.0910	7.7626	0.2560								
7	-0.0357	-0.0362	8.2477	0.3113								
8	0.0079	-0.0002	8.2716	0.4074								
9	-0.0970	-0.1201	11.866	0.2210								
10	0.0759	0.0635	14.074	0.1696								
11	0.0255	0.0143	14.324	0.2156								
12	-0.0225	-0.0568	14.519	0.2688								
13	0.0008	-0.0004	14.519	0.3383								

Although these tests give no reason to discard our ARIMA(0,1,1), the month-to-month changes in global temperature anomaly prove relatively hard to predict. Figure 12.17 employs the difference operator **D**. in a **graph** command to show the unimpressive fit between observed and predicted values. This model explains only about 20% of the variance in monthly differences.

```
. tsline D.ncdctemp Dncdchat, lcolor(blue red)
    xtitle("") xlabel(), grid gmax gmin) lw(medthick medium)
    ytitle("Monthly change in temperature, `=char(176)'C")
    ylabel(), grid gmin gmax) yline(0) legend(ring(0) position(7)
    row(2) label(1 "change in temp") label(2 "predicted change"))
```

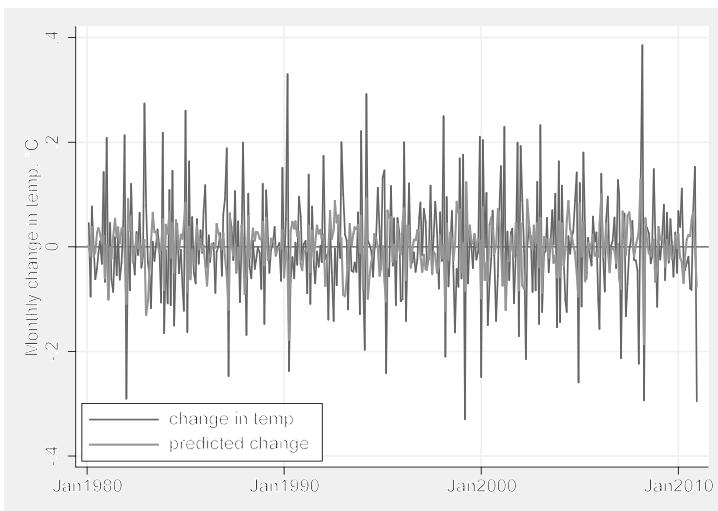


Figure 12.17

The plot of changes or first differences in Figure 12.17 bears little resemblance to the temperature anomalies themselves, seen in Figure 12.9. The contrast serves to emphasize that modeling first differences answers a different research question. In this example, a key feature of the temperatures record — the upward trend — has been set aside. The next section returns to the undifferenced anomalies and considers how the trend itself could be explained.

ARMAX Models

Earlier in this chapter we saw that an OLS regression of *ncdctemp* on four lagged predictors gave a good fit to observed temperature values (Figure 12.2), as well as physically plausible parameter estimates. A Durbin–Watson test found significant autocorrelation among the residuals, however, which undermines the OLS *t* and *F* tests. ARMAX (autoregressive moving average with exogenous variables) modeling provides a better approach for this problem.

A similar ARMAX model with one-month lagged predictors and ARIMA(1,0,1) disturbances proves to fit well.

```
. arima ncdctemp L1.aod L1.tsil L1.mei L1.co2anom, arima(1,0,1) nolog
ARIMA regression
Sample: 2 - 372                                         Number of obs      =      371
Log likelihood = 378.3487                               Wald chi2(6)      =     555.93
                                                       Prob > chi2      =     0.0000
```

ncdctemp	Coef.	OPG Std. Err.	z	P> z	[95% Conf. Interval]
ncdctemp	aod				
	L1.	-1.228967	.3855346	-3.19	0.001
	tsil	.0609574	.0173356	3.52	0.000
	L1.	.0533736	.0099622	5.36	0.000
	mei				
	L1.	.0104806	.0008328	12.58	0.000
	co2anom				
	L1.	-82.84698	23.68097	-3.50	0.000
	_cons				
ARMA	ar				
	L1.	.7119695	.0703746	10.12	0.000
	ma				
	L1.	-.3229314	.0944706	-3.42	0.001
	/sigma	.0872355	.0028313	30.81	0.000

Note: The test of the variance against zero is one sided, and the two-sided confidence interval is truncated at zero.

In this model y_t , or *ncdctemp* at time t , is a function of the lag-1 values of predictor variables x_1 through x_4 (*aod* through *co2anom*) and a disturbance (μ_t):

$$y_t = \beta_0 + \beta_1 x_{1,t-1} + \beta_2 x_{2,t-1} + \beta_3 x_{3,t-1} + \beta_4 x_{4,t-1} + \mu_t \quad [12.7]$$

The disturbance at time t (μ_t) is related to the lag-1 disturbance (μ_{t-1}) and to lag-1 and time t random errors (ϵ_{t-1} and ϵ_t):

$$\mu_t = \rho\mu_{t-1} + \theta\epsilon_{t-1} + \epsilon_t \quad [12.8]$$

Coefficients on all four predictors, and on the AR and MA terms, are statistically significant at $p < .01$ or better. *ncdctemp* itself is not stationary, and does not need to be for this analysis. Indeed, its nonstationarity is the focus of research. Residuals from a successful model, however, should resemble white noise, a covariance stationary process. That is the case here: a portmanteau *Q* test gives $p = .60$ for residuals up to lag 25).

```
. predict ncdchat2
. label variable ncdchat2 "Predicted from volcanoes,
    solar, ENSO & CO2"
. predict ncdrres2, resid
. label variable ncdrres2 "NCDC residuals from ARMAX model"
. corrgram ncdrres2, lags(25)
```

LAG	AC	PAC	Q	Prob>Q	-1	0	1	-1	0	1
					[Autocorrelation]		[Partial Autocor]			
1	-0.0133	-0.0136	.06661	0.7963						
2	0.0458	0.0471	.85498	0.6521						
3	-0.0089	-0.0084	.88473	0.8291						
4	0.0248	0.0229	1.1169	0.8916						
5	-0.0359	-0.0356	1.6043	0.9007						
6	-0.0658	-0.0710	3.245	0.7775						
7	-0.0060	-0.0035	3.2588	0.8601						
8	0.0116	0.0177	3.3097	0.9134						
9	-0.0949	-0.0979	6.7554	0.6626						
10	0.0832	0.0868	9.4103	0.4937						
11	0.0341	0.0433	9.8585	0.5432						
12	0.0052	-0.0105	9.869	0.6275						
13	0.0388	0.0457	10.45	0.6568						
14	0.0330	0.0288	10.873	0.6960						
15	-0.0110	-0.0273	10.92	0.7583						
16	-0.0381	-0.0288	11.485	0.7786						
17	-0.0871	-0.0861	14.449	0.6351						
18	-0.0384	-0.0537	15.028	0.6601						
19	-0.0180	0.0130	15.155	0.7127						
20	-0.0508	-0.0484	16.174	0.7058						
21	-0.0619	-0.0764	17.688	0.6686						
22	0.0339	0.0422	18.143	0.6975						
23	0.0557	0.0551	19.379	0.6790						
24	0.0779	0.0652	21.802	0.5911						
25	0.0457	0.0457	22.638	0.5987						

Figure 12.18 visualizes the close fit between data and the ARMAX model. The model explains about 77% of the variance in temperature anomalies. These ARMAX results support a general conclusion from our earlier OLS exploration: the multi-decade upward trend in temperatures cannot be explained without taking human drivers into account. The apparent slowdown in warming over the last decade, on the other hand, is better explained by natural drivers — La Niña and low solar irradiance.

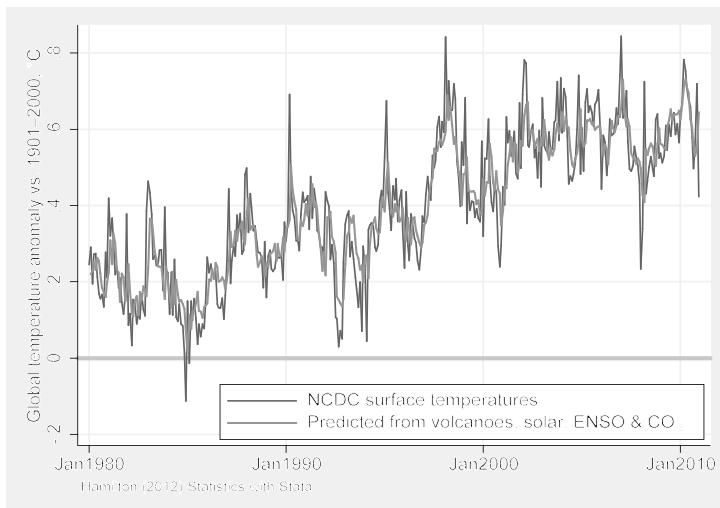


Figure 12.18

Up to this point we have dealt only with one of the three temperature indexes in *Climate.dta*. This index is produced by the National Climate Data Center (NCDC) of the U.S. National Oceanic and Atmospheric Administration (NOAA). NCDC calculates its index based on surface temperature measurements taken at thousands of stations around the globe. NASA's Goddard Institute for Space Studies produces its own temperature index (called GISTEMP), also based on surface station measurements but with better coverage of high-latitude regions.

The third temperature index in *Climate.dta* has a different basis. Researchers at the University of Alabama, Huntsville (UAH) calculate their global index from indirect satellite measurements of the upper troposphere, a region of Earth's atmosphere up to about 4 km altitude. The satellite-based estimates show greater month-to-month variability, and are more sensitive to El Niño and La Niña events. They provide an alternative perspective on global temperature trends, consistent with the surface records but with some differences in details. How do these differences manifest when we model *uahtemp* with the same ARMAX approach used with *ncdctemp*?

```
. arima uahtemp L1.aod L1.tsil L1.mei L1.co2anom, arima(1,0,1) nolog
```

ARIMA regression

Sample: 2 - 372	Number of obs	=	371
	Wald chi2(6)	=	601.88
Log likelihood = 299.2819	Prob > chi2	=	0.0000

uahtemp	Coef.	OPG Std. Err.	z	P> z	[95% Conf. Interval]
uahtemp	aod L1.	-2.38566	.9011263	-2.65	0.008
	tsil L1.	.0336446	.0289365	1.16	0.245
	mei L1.	.0663992	.0154607	4.29	0.000
	co2anom L1.	.0084778	.0016671	5.09	0.000
	_cons	-45.92205	39.52334	-1.16	0.245
ARMA	ar L1.	.8364133	.0421928	19.82	0.000
	ma L1.	-.3170064	.068849	-4.60	0.000
	/sigma	.1078988	.0040726	26.49	0.000
					.0999167 .115881

Note: The test of the variance against zero is one sided, and the two-sided confidence interval is truncated at zero.

Coefficients in the NCDC and UAH models, as well as NASA (not shown), all have the same sign, consistent with physical understanding. In all three models, CO₂ anomaly proves the strongest predictor of temperature, and MEI second strongest. MEI exerts relatively more influence in the UAH model, as expected with satellite data. Volcanic aerosols also have stronger effects on the UAH index, while solar irradiance exhibits weaker effects. Reflecting the higher variability of satellite-based estimates, the model explains only 70% of the variance

in *uahtemp*, compared with 77% for *ncdctemp*. However, the residuals pass tests for white noise ($p = .65$), and a plot of observed and predicted values shows a good visual fit (Figure 12.19).

```
. predict uahhat2
. label variable uahhat2 "Predicted from volcanoes, solar, ENSO & CO2"
. predict uahres2, resid
. label variable uahres2 "UAH residuals from ARMAX model"
. wntestq uahres2, lags(25)

Portmanteau test for white noise
```

Portmanteau (Q) statistic =	21.7197
Prob > chi2(25) =	0.6519

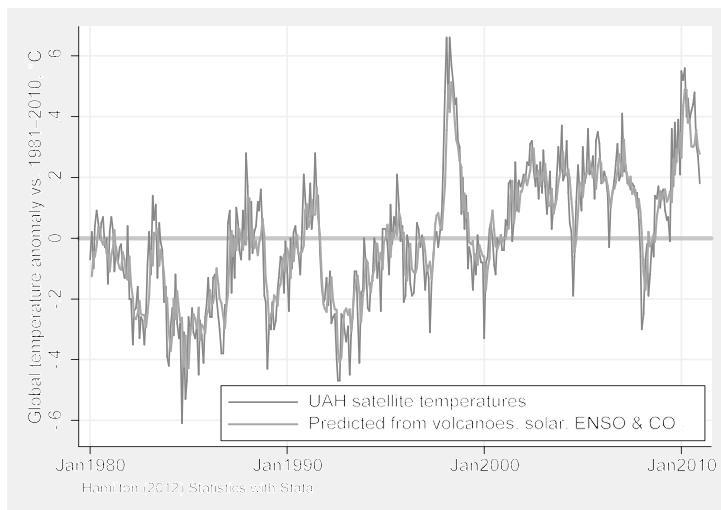


Figure 12.19

Satellite temperature records are a more recent development than weather-station records, so UAH temperature anomalies are calculated from a 1981–2010 baseline instead of 1901–2000 as done by NCDC. The recent baseline shifts the zero point for the UAH anomalies higher, as can be seen comparing Figure 12.19 with 12.18. Both series exhibit similar trends, however—about .16 °C/decade for NCDC (or NASA) over this period, or .15 °C/decade for UAH.

The conclusions from these simple ARMAX models generally agree with those from more sophisticated analyses by Foster and Rahmstorf (2011). Their approach involved searching for optimal lag specifications, in contrast to the arbitrary choice of 1-month lags for all predictors here. Foster and Rahmstorf also include trigonometric functions (a second-order Fourier series) to model residual annual cycles in the data. Time rather than CO₂ concentration was included among their predictors, to highlight temporal trends, but these trends are interpreted as human-caused. In principle those human causes could include other greenhouse gases besides CO₂, and other activities such as deforestation, land cover change or ozone depletion that are not captured by our *co2anom* variable, so a CO₂ causal interpretation may be overly simple. In practice, however, CO₂ has increased so directly with time (Figure 12.10) that using either time or CO₂ as a predictor will give similar results.

Multilevel and Mixed-Effects Modeling

Mixed-effects modeling is basically regression analysis allowing two kinds of effects: *fixed effects*, meaning intercepts and slopes meant to describe the population as a whole, just as in ordinary regression; and also *random effects*, meaning intercepts and slopes that can vary across subgroups of the sample. All of the regression-type methods shown so far in this book involve fixed effects only. Mixed-effects modeling opens a new range of possibilities for multilevel models, growth curve analysis, and panel data or cross-sectional time series.

Albright and Marinova (2010) provide a practical comparison of mixed-modeling procedures found in Stata, SAS, SPSS and R with the hierarchical linear modeling (HLM) software developed by Raudenbush and Bryck (2002; also Raudenbush et al. 2005). More detailed explanation of mixed modeling and its correspondences with HLM can be found in Rabe-Hesketh and Skrondal (2012). Briefly, HLM approaches multilevel modeling in several steps, specifying separate equations (for example) for level 1 and level 2 effects. In practice these equations cannot be estimated separately, so the software internally substitutes to form a single reduced equation for estimation. Mixed-effects modeling works directly with the reduced equation, giving it a “less multilevel” appearance than HLM even when both describe mathematically equivalent models. HLM effects at different levels can equivalently be represented as fixed or random effects within a single reduced equation. Rabe-Hesketh and Skrondal (2012:171) also note some cultural differences between HLM and mixed-modeling applications: “Papers using [HLM] tend to include more cross-level interactions and more random coefficients in the models (because the level-2 models look odd without residuals) than papers using for instance Stata.”

Three Stata commands provide the most general tools for multilevel and mixed-effects modeling. **xtmixed** fits linear models, like a mixed-effects counterpart to **regress**. Similarly, **xtmelogit** fits mixed-effects logit regression models for binary outcomes, like a generalization of **logit** or **logistic**; and **xtmepoisson** fits mixed-effects Poisson models for count outcomes, like a generalization of **poisson**. Stata also offers a number of more specialized procedures for conceptually related tasks. Examples include tobit, probit and negative binomial models with random intercepts; type **help xt** to see a complete list with links to details about each command. Many of these commands were first developed for use with panel or cross-sectional time series data, hence their common **xt** designation.

The **xtmixed**, **xtmelogit** and **xtmepoisson** procedures can be called either through typed commands or through menus,

Statistics > Multilevel mixed-effects models

Menus for other **xt** procedures are grouped separately under

Statistics > Longitudinal/panel data

The *Longitudinal/Panel Data Reference Manual* contains examples, technical details and references for mixed-effects and other **xt** methods. Luke (2004) provides a compact introduction to multilevel modeling. More extended treatments include books by Bickel (2007), McCulloch and Searle (2001), Raudenbush and Bryk (2002), and Verbeke and Molenberghs (2000). One particularly valuable resource for Stata users, *Multilevel and Longitudinal Modeling Using Stata* (Rabe-Hesketh and Skrondal 2012), describes not only the official Stata **xt** methods but also an unofficial program named **gllamm** that adds mixed-effect generalized linear modeling capabilities. Type **findit gllamm** for information on how to obtain and install the ado-files for this program.

Example Commands

. **xtmixed crime year || city: year**

Performs mixed-effects regression of *crime* on *year*, with random intercept and slope for each value of *city*. Thus, we obtain trends in crime rates, which are a combination of the overall trend (fixed effects), and variations on that trend (random effects) for each city.

. **xtmixed SAT parentcoll prepcourse grades || city: || school: grades**

Fits a hierarchical or multilevel mixed-effects model predicting students's SAT scores as a function of (1) fixed or whole-sample effects of whether the individual students' parent(s) graduated from college, whether the student took a preparation course, and student grades; (2) random intercepts representing the effect of the city in which they attend school; and (3) a random intercept and slope for the effect of individual students' grades, which could be different from one school to the next. Individual students (observations) are nested within schools, which are nested within cities. Note the order of mixed-effects parts in the command.

. **xtmixed y x1 x2 x3 || state: x3**

. **estimates store A**

. **xtmixed y x1 x2 x3 || state:**

. **estimates store B**

. **lrtest A B**

Conducts likelihood-ratio χ^2 test of null hypothesis of no difference in fit between the more complex model *A*, which includes a random slope on *x3*, and the simpler model *B* (*B* nested within *A*) that does not include a random slope on *x3*. This amounts to a test of whether the random slope on *x3* is statistically significant. The order of the two models specified in the **lrtest** command does not matter; if we had typed instead **lrtest B A**, Stata would still have correctly inferred that *B* is nested within *A*.

. **xtmixed y x1 x2 x3 || state: x2 x3, reml nocons cov(unstructured)**

Performs mixed-effects regression of *y* on fixed-effects predictors *x1*, *x2* and *x3*; also on random effects of *x2* and *x3* for each value of *state*. Obtains estimates by maximum restricted likelihood. The model should have no random intercept, and an unstructured covariance matrix in which random-effect variances and covariances all are estimated distinctly.

. **estat recov**

After **xtmixed**, displays the estimated variance-covariance matrix of the random effects.

. **predict re*, reffects**

After **xtmixed** estimation, obtains best linear unbiased predictions (BLUPs) of all random effects in the model. The random effects are stored as variables named *re1*, *re2* and so forth.

. **predict yhat, fitted**

After **xtmixed** estimation, obtains predicted values of *y*. To obtain predictions from the fixed-effects portion of the model only, type **predict yhat, xb**. Other **predict** options find standard errors of the fixed portion (**stdp**), and residuals (**resid**) or standardized residuals (**rstan**). To see a full list of **xtmixed** postestimation commands, with links to their syntax and options, type **help xtmixed postestimation**.

. **xtmelogit y x1 x2 || state:**

Performs mixed-effects logit regression of {0, 1} variable *y* on *x1* and *x2*, with random intercepts for each level of *state*.

. **predict phat**

After **xtmelogit** estimation, obtains predicted probabilities from the complete (fixed plus random) model. Type **help xtmelogit postestimation** to see other postestimation commands, as well as a complete list of options for **predict**, including Pearson residuals (**pearson**) and deviance residuals (**deviance**).

. **xtmepoisson accidents x1 x2 x3, exposure(persondays) || season:
|| port: , irr**

Estimates mixed-effects Poisson model for *accidents*, a count of accidents on fishing vessels. Fixed-effect predictors, characteristics of individual vessels, are *x1*, *x2* and *x3*. Exposure is measured by the number of person-days at sea for that vessel. We include random intercepts for each *season* or year, and for *port* city nested within seasons. Report fixed-effect coefficients as incident rate ratios (**irr**).

. **gllamm warming sex educ,i(region) family(binomial) link(ologit) adapt**

Performs generalized linear latent and mixed modeling — in this example a mixed-effects ordered logit regression of ordinal variable *warming* on fixed-effect predictors *sex* and *educ*. Includes random intercepts for each value of *region*, estimated by adaptive quadrature. The **family()** and **link()** options can specify other models including multinomial logit, probit and complementary log-log. **gllamm** is not an official Stata program, but available at no cost online. Type **findit gllamm** for information on how to download and install the necessary files. Rabe-Hesketh and Skrondal (2012) provide details and examples using **gllamm**.

Regression with Random Intercepts

To illustrate **xtmixed**, we begin with county-level data on votes in the 2004 presidential election (Robinson 2005). In this election, George W. Bush (receiving 50.7% of the popular vote) defeated John Kerry (48.3%) and Ralph Nader (0.4%). One striking aspect of this election was its geographical pattern: Kerry won states on the West coast, the Northeast, and around the Great Lakes, while Bush won everywhere else. Within states, Bush's support tended to be stronger in rural areas, whereas Kerry's votes concentrated more in the cities. Dataset *election_2004i* contains election results and other variables covering most U.S. counties. A categorical variable notes census divisions (*cendiv*), which divide the U.S. into 9 geographical areas. Variables include the total number of votes cast (*votes*), the percent for Bush (*bush*), logarithm of population density (*logdens*) as an indicator of rural-ness, and other variables for the percent of county population belonging to ethnic minorities (*minority*), or adults having college degrees (*colled*).

```
. use C:\data\election_2004i.dta, clear
. describe

Contains data from C:\data\election_2004i.dta
    obs:          3,054                               US counties -- 2004 election
    vars:          11                               (Robinson 2005)
    size:        219,888                           1 Jul 2012 14:35

    variable   storage      display      value      variable label
           name    type       format     label
    fips          long      %9.0g
    state         str20     %20s
    state2        str2      %9s
    region        byte      %9.0g
    cendiv        byte      %15.0g
    county        str24     %24s
    votes         float     %9.0g
    bush          float     %9.0g
    logdens       float     %9.0g
    minority      float     %9.0g
    colled        float     %9.0g

Sorted by: fips
```

The percent voting for Bush declined as population density increased, as visualized by the scatterplot and regression line in Figure 13.1. Each point represents one of the 3,054 counties.

```
. graph twoway scatter bush logdens, msymbol(Oh)
    || lfit bush logdens, lwidth(medthick)
```

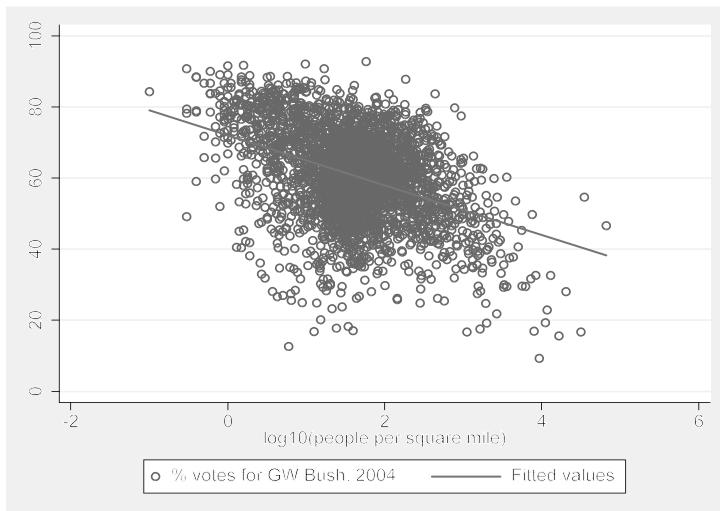


Figure 13.1

An improved version of this voting–density scatterplot appears in Figure 13.2. The logarithmic x-axis values have been relabeled (1 becomes “10”, 2 becomes “100,” and so forth) to make them more reader-friendly. Using *votes* as frequency weights for the scatterplot causes marker symbol area to be proportional to the total number of votes cast, visually distinguishing counties with small or large populations. Otherwise, analyses in this chapter do not make use of weighting. We focus here on patterns of county voting, rather than the votes of individuals within those counties.

```
. graph twoway scatter bush logdens [fw=votes], msymbol(Oh)
    || lfit bush logdens, lwidth(medthick)
    || , xlabel(-1 "0.1" 0 "1" 1 "10" 2 "100" 3 "1,000"
        4 "10,000", grid) legend(off)
    xtitle("Population per square mile")
    ytitle("Percent vote for GW Bush")
```

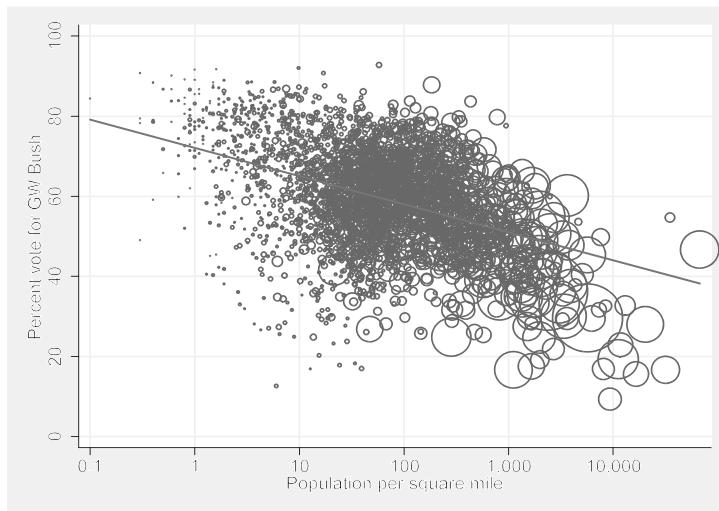


Figure 13.2

As Figure 13.2 confirms, the percent voting for George W. Bush tended to be lower in high-density, urban counties. It tended also to be lower in counties with substantial minority populations, or a greater proportion of adults with college degrees.

```
. regress bush logdens minority colled
```

Source	SS	df	MS	Number of obs = 3041
Model	122345.617	3	40781.8725	F(3, 3037) = 345.39
Residual	358593.826	3037	118.075017	Prob > F = 0.0000
Total	480939.443	3040	158.203764	R-squared = 0.2544
				Adj R-squared = 0.2537
				Root MSE = 10.866

bush	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
logdens	-5.457462	.3031091	-18.00	0.000	-6.051781 -4.863142
minority	-.251151	.0125261	-20.05	0.000	-.2757115 -.2265905
colled	-.1811345	.0334151	-5.42	0.000	-.246653 -.115616
_cons	75.78636	.5739508	132.04	0.000	74.66099 76.91173

In mixed-modeling terms, we just estimated a model containing only fixed effects — intercept and coefficients that describe the sample as a whole. The same fixed-effects model can be estimated using **xtmixed**, with similar syntax.

```
. xtmixed bush logdens minority colled
```

Mixed-effects ML regression			Number of obs	=	3041
Log likelihood = -11567.783			Wald chi2(3)	=	1037.53
			Prob > chi2	=	0.0000
bush	Coef.	Std. Err.	z	P> z 	[95% Conf. Interval]
logdens	-5.457462	.3029097	-18.02	0.000	-6.051154 -4.86377
minority	-.251151	.0125179	-20.06	0.000	-.2756856 -.2266164
colled	-.1811345	.0333931	-5.42	0.000	-.2465838 -.1156852
_cons	75.78636	.5735732	132.13	0.000	74.66217 76.91054
Random-effects Parameters		Estimate	Std. Err.	[95% Conf. Interval]	
sd(Residual)		10.85908	.1392419	10.58958	11.13545

Maximum likelihood (ML) is the default estimation method for **xtmixed**, but could be requested explicitly with the option **ml**. Alternatively, the option **reml** would call for maximum restricted likelihood estimation. See **help xtmixed** for a list of estimation, specification and reporting options.

The geographical pattern of voting seen in red state/blue state maps of this election are not captured by the fixed-effects model above, which assumes that the same intercept and slopes characterize all 3,041 counties in this analysis. One way to model the tendency toward different voting patterns in different parts of the country (and to reduce the problem of spatially correlated errors) is to allow each of the nine census divisions (New England, Middle Atlantic, Mountain, Pacific etc.) to have its own random intercept. Instead of the usual (fixed-effects) regression model such as

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \epsilon_i \quad [13.1]$$

we could include not only a set of β coefficients that describe all the counties, but also a random intercept u_0 , which varies from one census division to the next.

$$y_{ij} = \beta_0 + \beta_1 x_{1ij} + \beta_2 x_{2ij} + \beta_3 x_{3ij} + u_{0j} + \epsilon_{ij} \quad [13.2]$$

Equation [13.2] depicts the value of y for the i th county and the j th census division as a function of x_1 , x_2 and x_3 effects that are the same for all divisions. The random intercept u_{0j} , however, allows for the possibility that the mean percent voting for Bush is systematically higher or lower among the counties of some divisions. That seems appropriate with regard to U.S. voting, given the obvious geographical patterns. We can estimate a model with random intercepts for each census division by adding a new random-effects part to the **xtmixed** command:

```
. xtmixed bush logdens minority colled || cendiv:
```

Performing EM optimization:

Performing gradient-based optimization:

Iteration 0: Log likelihood = -11339.79
 Iteration 1: Log likelihood = -11339.79 (backed up)

Computing standard errors:

Mixed-effects ML regression	Number of obs	=	3041
Group variable: cendiv	Number of groups	=	9
	Obs per group:	min =	67
		avg =	337.9
		max =	616
Log likelihood = -11339.79	Wald chi2(3)	=	1161.96
	Prob > chi2	=	0.0000

bush	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
logdens	-4.52417	.3621775	-12.49	0.000	-5.234025 -3.814316
minority	-.3645394	.0129918	-28.06	0.000	-.3900029 -.3390758
colled	-.0583942	.0357717	-1.63	0.103	-.1285053 .011717
_cons	72.09305	2.294039	31.43	0.000	67.59682 76.58929

Random-effects Parameters	Estimate	Std. Err.	[95% Conf. Interval]
cendiv: Identity			
sd(_cons)	6.617135	1.600467	4.119006 10.63035
sd(Residual)	10.00339	.1284657	9.754742 10.25837

LR test vs. linear regression: chibar2(01) = 455.99 Prob >= chibar2 = 0.0000

The upper section of an **xtmixed** output table shows the fixed-effects part of our model. This model implies nine separate intercepts, one for each census division, but they are not directly estimated. Instead, the lower section of the table gives an estimated standard deviation of the random intercepts (6.62), along with a standard error (1.60) and 95% confidence interval for that standard deviation. Our model is

$$bush_{ij} = 72.09 - 4.52logdens_{ij} - .36minority_{ij} - .06colled_{ij} + u_{0j} + \epsilon_{ij} \quad [13.3]$$

If the standard deviation of u_0 appears significantly different from zero, we conclude that these intercepts do vary from place to place. That seems to be the case here — the standard deviation is more than four standard errors from zero, and its value is substantial (6.62 percentage points) in the metric of our dependent variable, percent voting for Bush. A likelihood-ratio test reported on the output's final line confirms that this random-intercept model offers significant improvement over a linear regression model with fixed effects only ($p \approx .0000$).

Although **xtmixed** does not directly calculate random effects, we can obtain the best linear unbiased predictions (BLUPS) of random effects through **predict**. The following commands create a new variable named *randint0*, containing the predicted random intercepts, and then graph each census division's random intercept in a bar chart (Figure 13.3).

```
. predict randint0, reffects
```

```
. graph hbar (mean) randint0, over(cendiv)
    ytitle("Random intercepts by census division")
```

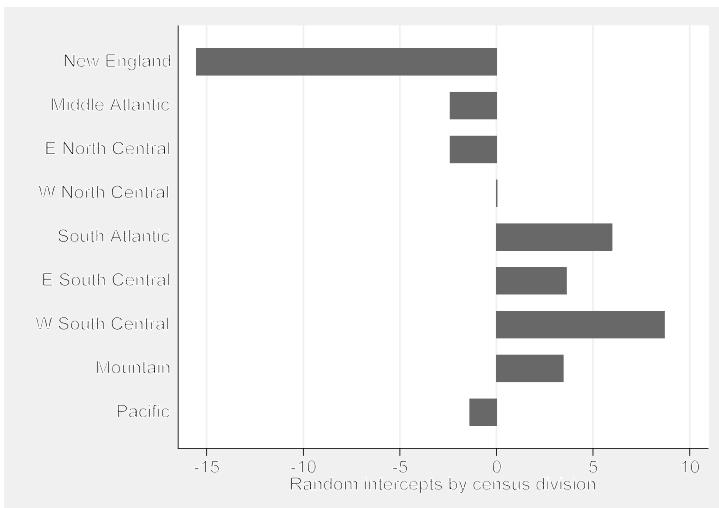
**Figure 13.3**

Figure 13.3 reveals that, at any given level of *logdens*, *minority* and *colled*, the percentage of votes going to Bush averaged about 15 points lower among New England counties, or more than 8 points higher in the W South Central (counties in Arkansas, Louisiana, Oklahoma and Texas), compared with the middle-of-the-road W North Central division.

Random Intercepts and Slopes

In Figure 13.2 we saw that, overall, the percentage of Bush votes tended to decline as population density increased. Our random-intercept model in the previous section accepted this generalization, while allowing intercepts to vary across regions. But what if the slope of the votes–density relationship also varies across regions? A quick look at scatterplots for each region (Figure 13.4) gives us grounds to suspect that it does.

```
. graph twoway scatter bush logdens, msymbol(Oh)
    || lfit bush logdens, lwidth(medthick)
    || , xlabel(-1(1)4, grid) ytitle("Percent vote for GW Bush")
    by(cendiv, legend(off) note(""))
```

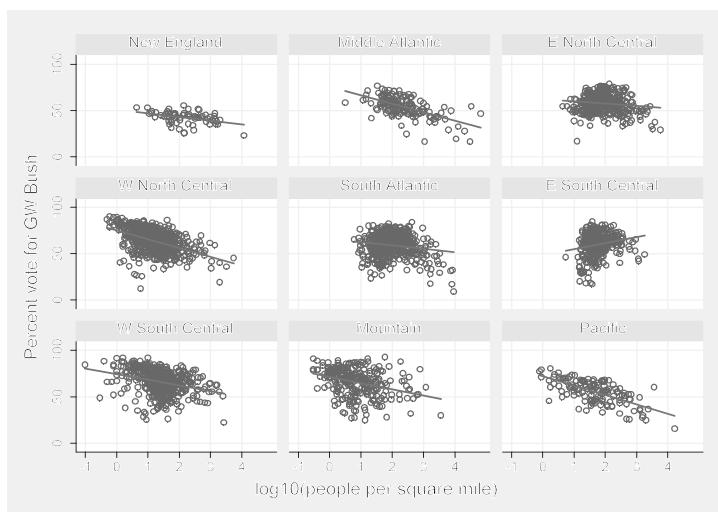


Figure 13.4

Bush votes decline most steeply with rising density in the Pacific and W North Central regions, but there appears to be little relationship in the E North Central, and even a positive effect in the E South Central. The negative fixed-effect coefficient on *logdens* in our previous model was averaging these downward, flat and upward trends together.

A mixed model including random slopes (u_{1j}) on predictor x_1 , and random intercepts (u_{0j}) for each of j groups could have the general form

$$y_{ij} = \beta_0 + \beta_1 x_{1ij} + \beta_2 x_{2ij} + \beta_3 x_{3ij} + u_{0j} + u_{1j} x_{1ij} + \epsilon_{ij} \quad [13.4]$$

To estimate such a model, we add the predictor variable *logdens* to the mixed-effect part of the **xtmixed** command.

```
. xtmixed bush logdens minority colled || cendiv: logdens
```

Performing EM optimization:

Performing gradient-based optimization:

```
Iteration 0: Log likelihood = -11298.734
Iteration 1: Log likelihood = -11298.734
```

Computing standard errors:

Mixed-effects ML regression Group variable: cendiv	Number of obs = 3041
	Number of groups = 9
	Obs per group: min = 67
	avg = 337.9
	max = 616
	Wald chi2(3) = 806.25
Log likelihood = -11298.734	Prob > chi2 = 0.0000

bush	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
logdens	-3.310313	1.114965	-2.97	0.003	-5.495605 -1.125021
minority	-.3616886	.0130709	-27.67	0.000	-.387307 -.3360702
colled	-.1173469	.0360906	-3.25	0.001	-.1880833 -.0466105
_cons	70.12095	2.955209	23.73	0.000	64.32885 75.91305

Random-effects Parameters	Estimate	Std. Err.	[95% Conf. Interval]	
cendiv: Independent				
sd(logdens)	3.113575	.8143897	1.86474	5.198768
sd(_cons)	8.5913	2.232214	5.162945	14.29619
sd(Residual)	9.825565	.1264176	9.580889	10.07649

LR test vs. Linear regression: chi2(2) = 538.10 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

As usual, the random effects are not directly estimated. Instead, the **xтmixed** table gives estimates of their standard deviations. The standard deviation for coefficients on log density is 3.11 — almost four standard errors (.81) from zero — suggesting that there exists significant division-to-division variation in the slope coefficients. A more definitive likelihood-ratio test will support this inference. To perform this test, we quietly re-estimate the intercept-only model, store those estimates as *A* (an arbitrary name) then re-estimate the intercept-and-slope model, store those estimates as *B*, and finally perform a likelihood-ratio test for whether *B* fits significantly better than *A*. In this example it does ($p \approx .0000$), so we conclude that adding random slopes brought significant improvement.

```
. quietly xtmixed bush logdens minority colled || cendiv:
. estimates store A
. quietly xtmixed bush logdens minority colled || cendiv: logdens
. estimates store B
. lrtest A B

Likelihood-ratio test                               LR chi2(1) =      82.11
(Assumption: A nested in B)                      Prob > chi2 =    0.0000
```

Note: The reported degrees of freedom assumes the null hypothesis is not on the boundary of the parameter space. If this is not true, then the reported test is conservative.

This **lrtest** output warns that it “assumes the null hypothesis is not on the boundary of the parameter space,” otherwise the reported test is conservative. A note at the bottom of the previous **xтmixed** output also stated that its likelihood-ratio test is conservative. Both notes refer to the same statistical issue. Variance cannot be less than zero, so a null hypothesis stating a variance equals zero does lie on the boundary of the parameter space. In this situation the reported likelihood-ratio test probability represents an *upper bound* (“conservative”) for the actual probability. **xтmixed** detects the situation automatically, but it is not practical for **lrtest** to do so, hence the “If ...” wording from the latter.

The previous model assumes that random intercept and slopes are uncorrelated, equivalent to adding a **cov(independent)** option specifying the covariance structure. Other possibilities

include **cov(unstructured)**, which would allow for a distinct, nonzero covariance between the random effects.

```
. xtmixed bush logdens minority colled
    || cendifv: logdens, cov(unstructured)

Performing EM optimization:

Performing gradient-based optimization:

Iteration 0:  log likelihood = -11296.31
Iteration 1:  log likelihood = -11296.31 (backed up)

Computing standard errors:

Mixed-effects ML regression                               Number of obs      =     3041
Group variable: cendifv                                Number of groups   =        9
                                                               Obs per group: min =       67
                                                               avg =      337.9
                                                               max =      616

Log likelihood = -11296.31                               Wald chi2(3)      =     799.68
                                                               Prob > chi2      =     0.0000



| bush     | Coef.     | Std. Err. | z      | P> z  | [95% Conf. Interval] |
|----------|-----------|-----------|--------|-------|----------------------|
| logdens  | -3.150009 | 1.169325  | -2.69  | 0.007 | -.441844 -.858175    |
| minority | -.3611161 | .0130977  | -27.57 | 0.000 | -.3867872 -.3354451  |
| colled   | -.1230445 | .0361363  | -3.41  | 0.001 | -.1938704 -.0522186  |
| _cons    | 69.85194  | 3.168479  | 22.05  | 0.000 | 63.64184 76.06204    |


| Random-effects Parameters | Estimate | Std. Err. | [95% Conf. Interval] |
|---------------------------|----------|-----------|----------------------|
| cendifv: Unstructured     |          |           |                      |
| sd(logdens)               | 3.282749 | .8547255  | 1.970658 5.468447    |
| sd(_cons)                 | 9.240389 | 2.402183  | 5.551459 15.3806     |
| corr(logdens,_cons)       | -.675152 | .1958924  | -.909687 -.1140964   |
| sd(Residual)              | 9.823658 | .1263468  | 9.579118 10.07444    |


LR test vs. linear regression: chi2(3) = 542.95 Prob > chi2 = 0.0000



Note: LR test is conservative and provided only for reference.


```

The estimated correlation between the random slope on *logdens* and the random intercept is $-.675$, more than three standard errors from zero. A likelihood-ratio test agrees that allowing for this correlation results in significant ($p = .0277$) improvement over our previous model.

```
. estimates store C
. lrtest B C

Likelihood-ratio test
(Assumption: B nested in C)                               LR chi2(1)      =      4.85
                                                               Prob > chi2      =     0.0277
```

The current model is

$$bush_{ij} = 69.85 - 3.15 logdens_{ij} - .36 minority_{ij} - .12 colled_{ij} + u_{0j} + u_{1j} logdens_{ij} + \epsilon_{ij} \quad [13.5]$$

So what are the slopes relating votes to density for each census division? Again, we can obtain values for the random effects (here named *randslo1* and *randint1*) through **predict**. Our dataset by now contains several new variables.

```
. predict randslo1 randint1, reffects
. describe

Contains data from C:\data\election_2004i.dta
obs: 3,054
vars: 17
size: 265,698
US counties -- 2004 election
(Robinson 2005)
6 Jul 2012 13:55
-----  

variable name storage type display format value label variable label
fips long %9.0g FIPS code
state str20 %20s State name
state2 str2 %9s State 2-letter abbreviation
region byte %9.0g region Region (4)
cendiv byte %15.0g division Census division (9)
county str24 %24s County name
votes float %9.0g Total # of votes cast, 2004
bush float %9.0g % votes for GW Bush, 2004
logdens float %9.0g log10(people per square mile)
minority float %9.0g % population minority
colled float %9.0g % adults >25 w/4+ years college
randint0 float %9.0g BLUP r.e. for cendiv: _cons
_est_A byte %8.0g esample() from estimates store
_est_B byte %8.0g esample() from estimates store
_est_C byte %8.0g esample() from estimates store
randslo1 float %9.0g BLUP r.e. for cendiv: logdens
randint1 float %9.0g BLUP r.e. for cendiv: _cons
-----  

Sorted by: fips
Note: dataset has changed since last saved
```

The random slope coefficients range from -5.66 for counties in the W North Central division, to +4.63 in the E South Central.

```
. table cendiv, contents(mean randslo1 mean randint1)
```

Census division (9)	mean(randslo1)	mean(randint1)
New England	1.760172	-18.79535
Middle Atlantic	-.0611391	-2.198096
E North Central	3.618166	-9.007434
W North Central	-5.65562	8.328082
South Atlantic	1.738479	2.875959
E South Central	4.633666	-4.289312
W South Central	-.8330051	10.8619
Mountain	-1.975749	7.232863
Pacific	-3.224968	4.991385

To clarify the relationship between votes and population density we could rearrange equation [13.5], combining the fixed and random slopes on *logdens*,

$$bush_{ij} = 69.85 + (u_{1j} - 3.15) \logdens_{ij} - .36 \text{minority}_{ij} - .12 \text{colled}_{ij} + u_{0j} + \epsilon_{ij} \quad [13.6]$$

In other words, the slope for each census division equals the fixed-effect slope for the whole sample, plus the random-effect slope for that division. Among Pacific counties, for instance, the combined slope is $-3.15 - 3.22 = -6.37$. The nine combined slopes are calculated and graphed in Figure 13.5.

```
. gen slope1 = randslo1 + _b[logdens]
. graph hbar (mean) slope1, over(cendiv)
    ytitle("Change in % Bush vote, with each tenfold increase
in density")
```

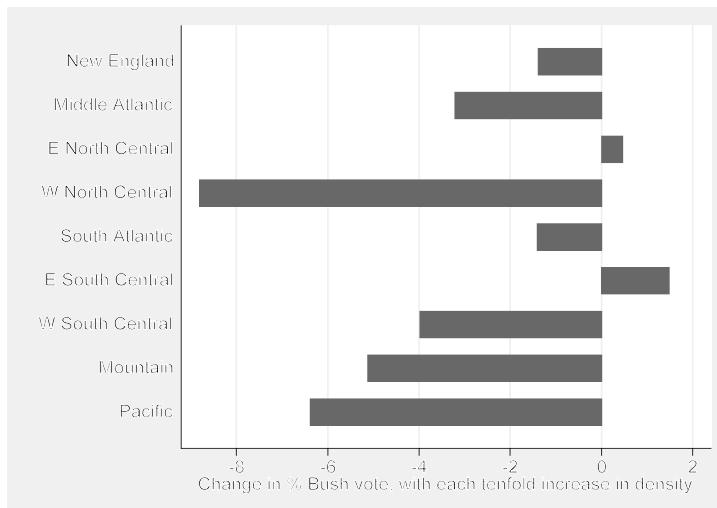


Figure 13.5

Figure 13.5 shows how the rural-urban gradient in voting behavior worked differently in different places. In counties of the W North Central, Pacific and Mountain regions, the percent voting for Bush declined most steeply as population density increased. In the E North Central and E South Central it went the other way — Bush votes increased slightly as density increased. The combined slopes in Figure 13.5 generally resemble those in the separate scatterplots of Figure 13.4, but do not match them exactly because our combined slopes (equation [13.6] or Figure 13.5) also adjust for the effects of minority and college-educated populations. In the next section we consider whether those effects, too, might have random components.

Multiple Random Slopes

To specify random coefficients on *logdens*, *minority* and *colled* we can simply add these variable names to the random-effects part of an **xtmixed** command. For later comparison tests, we save the estimation results with name *full*. Some of the iteration details have been omitted in the following output.

```
. xtmixed bush logdens minority colled
|| cendiv: logdens minority colled
```

Performing EM optimization:

Performing gradient-based optimization:

Iteration 0: log likelihood = -11184.804
 Iteration 1: log likelihood = -11184.804

Computing standard errors:

Mixed-effects ML regression	Number of obs	=	3041
Group variable: cendiv	Number of groups	=	9
	Obs per group: min	=	67
	avg	=	337.9
	max	=	616
Log likelihood = -11184.804	Wald chi2(3)	=	52.49
	Prob > chi2	=	0.0000

bush	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
logdens	-2.717128	1.373684	-1.98	0.048	-5.409499 -.0247572
minority	-.3795605	.0560052	-6.78	0.000	-.4893286 -.2697924
colled	-.1707863	.1727742	-0.99	0.323	-.5094175 .167845
_cons	70.86653	3.435918	20.63	0.000	64.13225 77.6008

Random-effects Parameters	Estimate	Std. Err.	[95% Conf. Interval]
cendiv: Independent			
sd(logdens)	3.868421	.9832899	2.350564 6.366421
sd(minority)	.153172	.0439569	.0872777 .2688161
sd(colled)	.5032414	.1241234	.310334 .8160625
sd(_cons)	10.01157	2.547813	6.079707 16.48625
sd(Residual)	9.375994	.1209753	9.141859 9.616124

LR test vs. linear regression: chi2(4) = 765.96 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

. estimates store full

Taking the *full* model as our baseline, likelihood-ratio tests establish that the random coefficients on *logdens*, *minority* and *colled* each have statistically significant variation, so these should be kept in the model. For example, to evaluate the random coefficients on *colled* we quietly estimate a new model without them (*nocolled*), then compare that model with *full*. The *nocolled* model fits significantly worse ($p \approx .0000$) than the *full* model seen earlier.

```
. quietly xtmixed bush logdens minority colled
    || cendiv: logdens minority
. estimates store nocolled
. lrtest nocolled full

Likelihood-ratio test                               LR chi2(1) =   197.33
(Assumption: nocolled nested in full)             Prob > chi2 =  0.0000
```

Note: The reported degrees of freedom assumes the null hypothesis is not on the boundary of the parameter space. If this is not true, then the reported test is conservative.

Similar steps with two further models (*nologdens* and *nominority*) and likelihood-ratio tests show that the *full* model also fits significantly better than models without either a random coefficient on *logdens* or one on *minority*.

```
. quietly xtmixed bush logdens minority colled
    || cendifv: minority colled
. estimates store nologdens
. lrtest nologdens full

Likelihood-ratio test                               LR chi2(1) =     124.87
(Assumption: nologdens nested in full)           Prob > chi2 =     0.0000

Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.

. quietly xtmixed bush logdens minority colled
    || cendifv: logdens colled
. estimates store nominority
. lrtest nominority full

Likelihood-ratio test                               LR chi2(1) =     38.76
(Assumption: nominority nested in full)           Prob > chi2 =     0.0000

Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
```

We could investigate the details of all these random effects, or the combined effects they produce, through calculations along the lines of those shown earlier for Figure 13.5.

Mixed-modeling research often focuses on the fixed effects, with random effects included to represent heterogeneity in the data, but not of substantive interest. For example, our analysis thus far has demonstrated that population density, percent minority and percent college educated predict county voting patterns nationally, even after adjusting for regional differences in mean votes and for regional effects of density and college grads. On the other hand, random effects might themselves be quantities of interest. To look more closely at how the relationship between voting and percent college graduates (or percent minority, or log density) varies across census divisions, we can predict the random effects and from these calculate total effects. These steps are illustrated for the total effect of *colled* in our full model below, and graphed in Figure 13.6.

```
. quietly xtmixed bush logdens minority colled
    || cendifv: logdens minority colled
. predict relogdens reminority recolled re_cons, reffects
. describe relogdens-re_cons

variable   storage   display   value
name      type      format    label      variable label
-----+
relogdens  float     %9.0g      BLUP r.e. for cendifv: logdens
reminority float     %9.0g      BLUP r.e. for cendifv: minority
recolled   float     %9.0g      BLUP r.e. for cendifv: colled
re_cons    float     %9.0g      BLUP r.e. for cendifv: _cons

. generate tecolled = recolled + _b[colled]
. label variable tecolled "random + fixed effect of colled"
. table cendifv, contents(mean recolled mean tecolled)
```

Census division (9)	mean(recolled)	mean(tecolled)
New England	-.2411761	-.4119623
Middle Atlantic	-.0849539	-.2557401
E North Central	-.0682224	-.2390087
W North Central	.4122158	.2414296
South Atlantic	-.1243222	-.2951084
E South Central	.4856058	.3148195
W South Central	.9054216	.7346353
Mountain	-.7355713	-.9063575
Pacific	-.5489974	-.7197837

```
. graph hbar (mean) tecolled, over(cendiv)
    ytitle("Change in % Bush vote, per 1% increase in
college graduates")
```

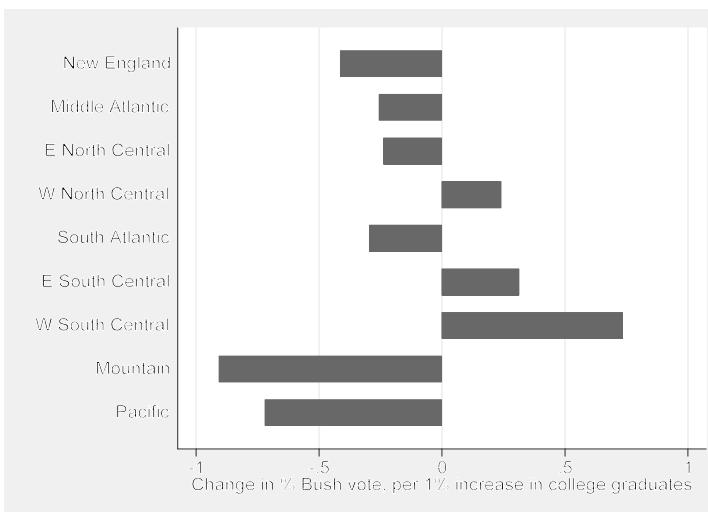


Figure 13.6

Figure 13.6 visualizes the reason our model was significantly improved ($p \approx .0000$) by including a random slope for *colled*. The total effects of *colled* on Bush votes range from substantially negative (percent Bush votes lower among counties with more college graduates) in the Mountain (−.91) and Pacific (−.72) census divisions, through negligible in the South Atlantic or W North Central, to substantially positive (+.73) in the W South Central where Bush votes were substantially higher in counties with more graduates — controlling for population density, percent minority and other regional effects. If we estimate only a fixed effect for *colled*, our model effectively would average these negative, near zero, and positive random effects of *colled* into one weakly negative fixed coefficient, specifically the −.18 coefficient in the two fixed-effects regressions that started off this chapter.

The examples seen so far were handled successfully by **xtmixed**, but that is not always the case. Mixed-model estimation can fail to converge for a variety of different reasons, resulting in repeatedly “nonconcave” or “backed-up” iterations, or error messages about Hessian or standard

error calculations. The *Longitudinal/Panel Data Reference Manual* discusses how to diagnose and work around convergence problems. A frequent cause seems to be models with near-zero variance components, such as random coefficients that really do not vary much, or have low covariances. In such cases the offending components can reasonably be dropped.

Nested Levels

Mixed-effects models can include more than one nested level. The counties of our voting data, for example, are nested not only within census divisions, but also within states that are nested within census divisions. Might there exist random effects not only at the level of census divisions, but also at the smaller level of states? The **xtmixed** command allows for such hierarchical models. Additional random-effects parts are added to the command, with successively smaller (nested) units to the right. The following analysis specifies random intercepts and slopes on all three predictors for each census division, and also random intercepts and slopes on percent college graduates, *colled*, for each state.

```
. xtmixed bush logdens minority colled
|| cendiv: logdens minority colled
|| state: colled
```

Performing EM optimization:

Performing gradient-based optimization:

```
Iteration 0:  log likelihood = -10719.828
Iteration 1:  log likelihood = -10719.821
Iteration 2:  log likelihood = -10719.821
```

Computing standard errors:

Mixed-effects ML regression	Number of obs	=	3041
-----------------------------	---------------	---	------

Group Variable	No. of Groups	Observations per Group		
		Minimum	Average	Maximum
cendiv	9	67	337.9	616
state	49	1	62.1	254

Log likelihood = -10719.821	Wald chi2(3)	=	68.85
	Prob > chi2	=	0.0000

bush	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
logdens	-2.473536	.996434	-2.48	0.013	-4.426511 -.5205616
minority	-.4067648	.0533714	-7.62	0.000	-.5113709 -.3021588
colled	-.1787849	.1298313	-1.38	0.168	-.4332495 .0756797
_cons	71.13208	3.048286	23.34	0.000	65.15755 77.10661

Random-effects Parameters	Estimate	Std. Err.	[95% Conf. Interval]	
cendiv: Independent				
sd(logdens)	2.703845	.7727275	1.544252	4.734186
sd(minority)	.1465435	.0428326	.0826365	.2598728
sd(colled)	.3683903	.0962733	.220729	.6148326
sd(_cons)	8.416873	2.417524	4.793643	14.77869
state: Independent				
sd(colled)	.1305727	.039009	.0727032	.2345047
sd(_cons)	5.883451	.7431715	4.593166	7.536196
sd(Residual)	7.863302	.1027691	7.664436	8.067328

LR test vs. linear regression: chi2(6) = 1695.92 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

At first glance, all of the random effects at both *cendiv* and *state* level in this output appear significant, judging from their standard errors and confidence intervals. The standard deviation of state-level random coefficients on *colled* (.13) is smaller than the standard deviation of corresponding census division-level coefficients (.37), but both are substantial relative to the fixed-effect coefficient on *colled* (-.18). Our confidence interval for the state-level coefficient ranges from .07 to .23. A likelihood-ratio test indicates that this model (here named *state*) with state-level random intercepts and slopes fits much better than our earlier model (*full*), which had only census division-level random intercepts and slopes.

```
. estimates store state
. lrtest full state

Likelihood-ratio test                               LR chi2(2) =    929.97
(Assumption: full nested in state)                Prob > chi2 =  0.0000
```

Note: The reported degrees of freedom assumes the null hypothesis is not on the boundary of the parameter space. If this is not true, then the reported test is conservative.

As before, we can predict the random effects, then use these to calculate and graph the total effects. For *colled*, we now have random effects for 49 different states. Box plots work well to show their distribution (Figure 13.7), which follows the general pattern of census division effects seen earlier in Figure 13.6, but now also within-division variation. Indiana (E North Central) and Oklahoma (W South Central) plot as outliers, because each is unusual within its division.

```
. predict re*, reffects
. describe re1-re6

variable   storage      display      value
          name       type        format     label      variable label
re1        float        %9.0g      BLUP r.e. for cendiv: logdens
re2        float        %9.0g      BLUP r.e. for cendiv: minority
re3        float        %9.0g      BLUP r.e. for cendiv: colled
re4        float        %9.0g      BLUP r.e. for cendiv: _cons
re5        float        %9.0g      BLUP r.e. for state: colled
re6        float        %9.0g      BLUP r.e. for state: _cons

. gen tecolled2 = re3 + re5 + _b[colled]
. label variable tecolled2
      "cendiv + state + fixed effect of % college grads"
```

```
. graph hbox tecolled2, over(cendiv) yline(-.16)
    marker(1, mlabel(state))
```

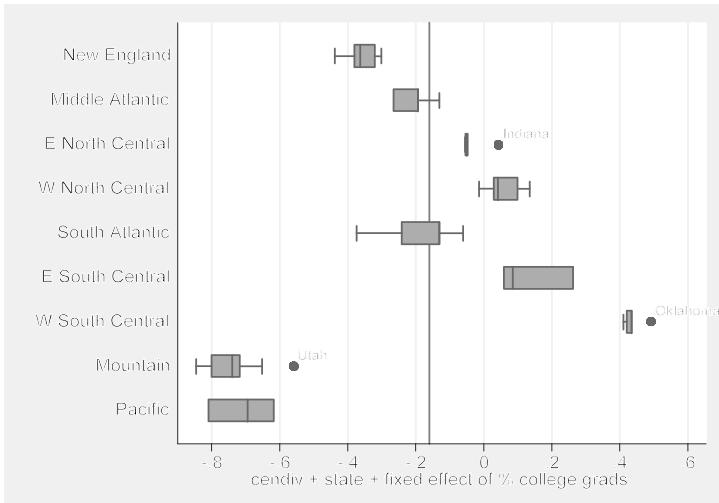


Figure 13.7

For other **xtmixed** postestimation tools, type **help xtmixed postestimation**. The *Longitudinal/Panel Data Reference Manual*, and also Rabe-Hesketh and Skrondal (2012), present further applications of mixed modeling such as blocked-diagonal covariance structures and crossed-effects models.

Repeated Measurements

Dataset *attract2.dta* describes an unusual experiment carried out at a college undergraduate party, where some drinking apparently took place. In this experiment, 51 college students were asked to individually rate the attractiveness, on a scale from 1 to 10, of photographs of men and women unknown to them. The rating exercise was repeated by each participant, given the same photos shuffled in random order, on four occasions over the course of the evening. Variable *ratemale* is the mean rating each participant gave to all the male photos in one sitting, and *ratefem* is the mean rating given to female photos. *gender* records the subject's own gender, and *bac* his or her blood alcohol content as measured by a breathalyzer.

```
. use C:\data\attract2.dta, clear
. describe
```

Contains data from C:\data\attract2.dta				Perceived attractiveness and drinking -- DC Hamilton (2003)	
obs:	204	vars:	7	1 Jul 2012 14:36	
size:	3,876				
variable	storage type	display format	value label	variable label	
id	byte	%9.0g		Participant number	
gender	byte	%9.0g	gender	Gender	
bac	float	%9.0g		Blood alcohol content	
single	byte	%9.0g	single	Relationship status single	
drinkfrq	float	%9.0g		Days drinking in previous week	
ratefem	float	%9.0g		Rated attractiveness of females	
ratemale	float	%9.0g		Rated attractiveness of males	

Sorted by: id bac

The research hypothesis involves a “beer goggles” effect: do strangers become more attractive as subjects consume more alcohol? In terms of this experiment, is there a positive relationship between blood alcohol content and the attractiveness ratings given to the photographs? If so, does that relationship vary with the gender of subject or photograph?

Although the data contain 204 observations, these represent only 51 individual participants. It seems reasonable that individual participants might differ in their tendency to give relatively high or low ratings, and in their reaction to alcohol. A mixed-effects model with random intercepts and slopes can accommodate these likely complications. This situation differs from the preceding election example in that individual random intercepts and slopes will not be substantively interesting, since they describe anonymous individuals. Rather, they are data or experimental-design features for which we need to adjust when testing the main hypotheses.

Using factor-variable notation, the following **xtmixed** command specifies a model in which mean ratings given to photographs of female faces (*ratefem*) are predicted by fixed effects of indicator variable *gender*, continuous variable *bac*, and their interaction. In addition, the model includes random intercepts and slopes for *bac*, which could vary across subjects.

. xtmixed ratefem i.gender##c.bac id: bac, nolog						
Mixed-effects ML regression		Number of obs = 204				
Group variable: id		Number of groups = 51				
		Obs per group: min = 4				
		avg = 4.0				
		max = 4				
Log Likelihood = -170.9156		Wald chi2(3) = 56.95				
		Prob > chi2 = 0.0000				
ratefem	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
1.gender	-.6280836	.3203292	-1.96	0.050	-1.255917	-.0002499
bac	3.433733	.5231428	6.56	0.000	2.408392	4.459074
gender##c.bac	-1.154182	.9270306	-1.25	0.213	-2.971128	.6627648
1						
_cons	6.442059	.1903235	33.85	0.000	6.069032	6.815086

Random-effects Parameters	Estimate	Std. Err.	[95% Conf. Interval]	
id: Independent				
sd(bac)	1.621421	.6564771	.7332693	3.585323
sd(_cons)	1.056773	.1087889	.8636849	1.293029
sd(Residual)	.3371602	.02408	.2931186	.387819
LR test vs. linear regression:		chi2(2) = 279.98	Prob > chi2 = 0.0000	

Note: LR test is conservative and provided only for reference.

These results for female photographs support the beer goggles hypothesis: attractiveness ratings of female faces increase with blood alcohol content. Gender has a barely-significant effect: women gave slightly lower ratings to female faces. The *gender*×*bac* interaction is not significant. However, both the random intercept and the random slope on *bac* exhibit significant variation, indicating substantial person-to-person differences both in the mean ratings given, and in how alcohol affected those ratings.

margins and **marginsplot** commands help to visualize these results. Their outcome is not shown here, but will be combined with the next analysis to form Figure 13.8.

```
. margins, at(bac = (0(.2).4) gender = (0 1)) vsquish
. marginsplot, title("Female photos") ytitle("") xtitle("") noci
    legend(position(11) ring(0) row(2)
        title("Gender", size(medsmall)))
    ylabel(4(1)8, grid gmin gmax)
    plot1opts(lpattern(solid) msymbol(T))
    plot2opts(lpattern(dash) msymbol(Oh)) saving(fig13_08RF)
```

A second **xtmixed** command models the ratings of male photographs (*ratemal*):

```
. xtmixed ratemal i.gender##c.bac || id: bac, nolog

Mixed-effects ML regression
Group variable: id                                         Number of obs      =      201
                                                               Number of groups   =       51
                                                               Obs per group: min =         3
                                                               avg =       3.9
                                                               max =         4

Log likelihood = -221.83425                               Wald chi2(3)      =     32.74
                                                               Prob > chi2      =     0.0000

                                         Coef.  Std. Err.      z  P>|z|  [95% Conf. Interval]
ratemale
1.gender                                2.011293  .3742453    5.37  0.000    1.277786  2.744801
bac                                     .6401159  .7601351    0.84  0.400   -.8497215  2.129953
gender##c.bac
1                                     .6055665  1.328251    0.46  0.648   -1.997758  3.208891
_cons                                 3.946884  .2224468   17.74  0.000    3.510897  4.382872
```

Random-effects Parameters	Estimate	Std. Err.	[95% Conf. Interval]	
id: Independent	sd(bac)	2.738856	.535468	1.867036
	sd(_cons)	1.223205	.1284553	.9956575
sd(Residual)	.4408696	.0278099	.389598	.4988886

LR test vs. linear regression: $\text{chi2}(2) = 255.98$ Prob > chi2 = 0.0000

Note: **LR test** is conservative and provided only for reference.

There does not seem to be a beer goggles effect on how male photographs are rated. On the other hand, gender exhibits a strong fixed effect. The interaction of *gender* and *bac* is not significant, but again we see significant variation in the random intercept and slope.

```
. margins, at(bac = (0(.2).4) gender = (0 1)) vsquish
. marginsplot, title("Male photos") ytitle("") xtitle("") noci
    legend(position(11) ring(0) row(2)
          title("Gender", size(medsmall)))
    ylabel(4(1)8, grid gmin gmax)
    plot1opts(lpattern(solid) msymbol(T))
    plot2opts(lpattern(dash) msymbol(O)) saving(fig13_08RM)
```

Figure 13.8 combines the two **marginsplot** graphs for direct comparison.

```
. graph combine fig13_08RF.gph fig13_08RM.gph
    , imargin(vsmall) l1("Mean attractiveness rating")
    b2("Blood alcohol content")
```

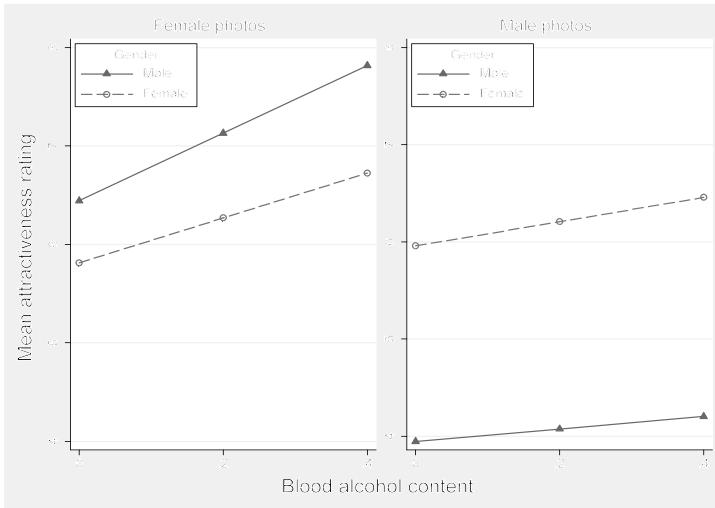


Figure 13.8

The left panel in Figure 13.8 visualizes the significant alcohol effect on ratings of female photographs. Male subjects gave somewhat higher ratings, and they appear slightly more affected by drink. The right panel depicts a different situation for ratings of male photographs.

Female subjects gave notably higher ratings, but neither male nor female subjects' ratings changed much as they drank.

Cross-Sectional Time Series

This section applies **xtmixed** to a different kind of multilevel data: cross-sectional time series. Dataset *Alaska_regions.dta* contains time series of population for each of the 27 boroughs, municipalities or census areas that together make up the state of Alaska. These 27 regions are a fragment from the pan-Arctic human-dimensions database framework described by Hamilton and Lammers (2011). In *Alaska_regions.dta*, a dummy variable named *large* marks the five most populous Alaska regions, those having 2011 populations greater than 20,000. The other 22 regions are predominantly rural, and their smaller populations might be widely dispersed. The Northwest Arctic borough, for example, covers a geographic area larger than the state of Maine, but had a 2011 population of fewer than 8,000 people. For each of the 27 regions there are multiple years of data, ranging from 1969 to 2011 but with many missing values. Thus, for each variable there are 27 parallel but often incomplete time series.

```
. use C:\data\Alaska_regions.dta, clear
. describe

Contains data from C:\data\Alaska_regions.dta
obs: 852
vars: 7
size: 44,304
Alaska regions population
 1969-2011
1 Jul 2012 14:37

variable name  storage type   display format value label
regionname    str34
regioncode    float
year          int
pop           double %12.0g
large         byte
year0         byte %9.0g
year2         int %9.0g
large          large
Region name
AON-SI region code
Year
Population in thousands
Regions 2011 population > 20,000
years since 1968
years0 squared

Sorted by: regionname year
```

During the first half of the time period covered by these data, population grew substantially in many parts of rural Alaska. In more recent years, however, the rate of growth leveled off, and populations in some areas declined. The trends are relevant to discussions about sustainable economic development for these places, and also for their cultural importance to the Alaska Native populations who live there.

Because population trends have not simply gone upwards, they cannot realistically be modeled as a linear function of *year*. The mixed model below represents population trends as a quadratic function, regressing population in thousands (*pop*) on years since 1968 (*year0*) and also on *year0* squared. We allow for fixed (β) and random (u_i) intercepts and slopes on both terms. The 5 large-population regions are set aside for these analysis, in order to focus on rural Alaska.

$$\text{population}_{ij} = \beta_0 + \beta_1 \text{year0}_{ij} + \beta_2 \text{year0}^2_{ij} + u_{0j} + u_{1j} \text{year0}_{ij} + u_{2j} \text{year0}^2_{ij} + \epsilon_{ij} \quad [13.7]$$

```

. keep if large == 0
. xtmixed pop year0 year2 || areaname: year0 year2

Performing EM optimization:

Performing gradient-based optimization:

Iteration 0:  log likelihood = -457.61229
Iteration 1:  log likelihood = -457.61196
Iteration 2:  log likelihood = -457.61196

Computing standard errors:

Mixed-effects ML regression
Number of obs      =      639
Group variable: regionname   Number of groups =       22
                                         Obs per group: min =        22
                                                               avg =     29.0
                                                               max =      40
                                         Wald chi2(2)      =      1.22
Log likelihood = -457.61196          Prob > chi2      =    0.5424



| pop   | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |
|-------|-----------|-----------|-------|-------|----------------------|
| year0 | .0615239  | .0783008  | 0.79  | 0.432 | -.0919428 .2149906   |
| year2 | -.0008945 | .0010787  | -0.83 | 0.407 | -.0030087 .0012197   |
| _cons | 5.457939  | 1.342116  | 4.07  | 0.000 | 2.82744 8.088438     |


| Random-effects Parameters | Estimate | Std. Err. | [95% Conf. Interval] |
|---------------------------|----------|-----------|----------------------|
| regionname: Independent   |          |           |                      |
| sd(year0)                 | .3563579 | .0619934  | .2534009 .5011464    |
| sd(year2)                 | .004861  | .0008663  | .0034278 .0068933    |
| sd(_cons)                 | 6.145796 | 1.04392   | 4.405485 8.573587    |
| sd(Residual)              | .3524305 | .0108473  | .3317988 .3743451    |


LR test vs. linear regression: chi2(3) = 2703.76 Prob > chi2 = 0.0000


```

Note: LR test is conservative and provided only for reference.

All the random effects show significant variation from place to place. The fixed-effect coefficients on *year0* and *year2*, on the other hand, do not differ significantly from zero — indicating the lack of an overall trend across these 22 regions. Graphing predicted population (with a median-spline curve) and actual population against calendar year helps to visualize the details of area-to-area variation and why **xtmixed** finds no overall trend (Figure 13.9). In some areas the population grew steadily, while in others the direction of growth or decline reversed. The model does a decent job of smoothing past some visible discontinuities in the data, such as the Aleutians West population drop that followed downsizing of a naval air station in 1994, or the North Slope gain that followed inclusion of employees at remote work sites in the 2010 Census.

```

. predict yhat, fitted
. graph twoway scatter pop year, msymbol(Oh)
    || mspline yhat year, lwidth(medthick) bands(50)
    || , by(regionname, note("") legend(off))
    ylabel(0(5)20, angle(horizontal)) xtitle("")
    ytitle("Population in 1,000s") xlabel(1970(10)2010, grid)

```

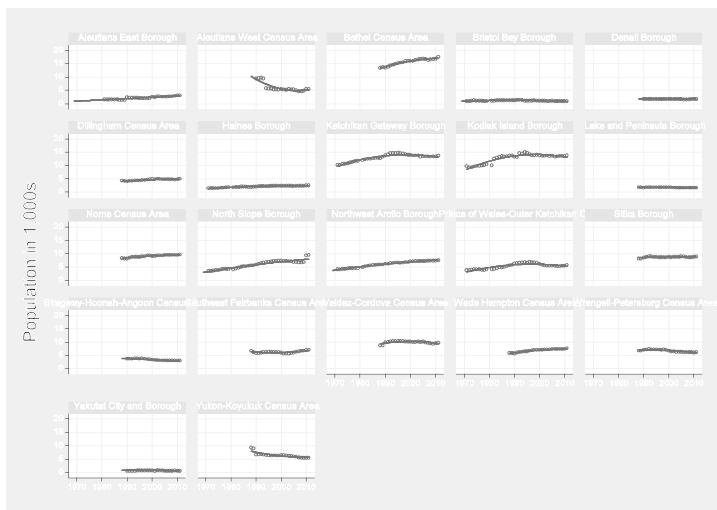


Figure 13.9

A more substantive analysis using mixed-effects regression to model relationships involving multiple time series appears in a paper on population, climate and electricity use in towns and villages of Arctic Alaska (Hamilton et al. 2011). Dataset *Alaska_places.dta* contains the core data for this analysis, consisting of annual time series for each of 42 towns and villages in Arctic Alaska, all within five of the census regions or boroughs graphed in Figure 13.9 above. Variables include community population, kilowatt hours of electricity sold, the average rate charged for electricity (in constant 2009 dollars), and estimates of the summer temperature and precipitation around that location each year. The paper supplies information about variable definitions, data sources and the context for this analysis.

<pre>. use C:\data\Alaska_places.dta, clear . describe</pre>				
<i>Contains data from C:\data\Alaska_places.dta</i>				
obs:	742	Population, climate & electricity use in the Arctic (Hamilton 2011)		
vars:	12	6 Jul 2012 19:34		
size:	57,876			
variable	name	storage type	display format	value label
regionname		str24	%24s	Region name
regioncode		long	%9.0g	AON-SI region code
place		str21	%21s	Place name
placecode		byte	%21.0g	placecode
year		int	%ty	Place code (labeled)
pop		int	%12.0g	Year
logpop		float	%9.0g	Population--est. Jul 1/Census Apr 1
kwhsold20		float	%9.0g	log10(pop)
logkwhsold		float	%9.0g	kwh sold adjusted if 9-11 months, millions
rateres09		float	%9.0g	log10(kwhsold20)
				av. res. rate 2009\$ = rateres*cpianc09

```

fsumtempD      float %9.0g          UDel FY summer (L1.Jul-Sep &
fsumprecD     float %9.0g          May-Jun) temp
                           UDel FY summer (L1.Jul-Sep &
                           May-Jun) prec

```

Sorted by: placecode year

These data have been declared as panel through an `xtset` command that specified `placecode` as the panel variable and `year` as the time variable:

```

. xtset placecode year
    panel variable: placecode (unbalanced)
    time variable: year, 1990 to 2008, but with gaps
        delta: 1 year

```

The mixed-effect analysis below models the kilowatt hours of electricity sold in each community, each year, as a function of population, electricity rate in constant 2009 dollars, summer temperature and precipitation, and year. The model includes random effects for population, and first-order autoregressive errors. Reasoning behind this particular specification, and tests for the robustness of modeling results, are given in the paper; the short presentation here just aims to show what such an analysis looks like.

```

. xtmixed logkwhsold logpop rateres09 fsumtempD fsumprecD year
|| placecode: logpop, nocons residual(ar 1, t(year)) nolog reml

```

Note: time gaps exist in the estimation data

Mixed-effects REML regression	Number of obs =	742
Group variable: placecode	Number of groups =	42
	Obs per group: min =	12
	avg =	17.7
	max =	19

Log restricted-likelihood = 1008.232	Wald chi2(5) = 359.12
	Prob > chi2 = 0.0000

logkwhsold	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
logpop	.7086405	.0716509	9.89	0.000	.5682073 .8490737
rateres09	-.0011494	.0005259	-2.19	0.029	-.0021801 -.0001187
fsumtempD	-.0038939	.0018784	-2.07	0.038	-.0075755 -.0002123
fsumprecD	.000272	.0001416	1.92	0.055	-5.57e-06 .0005495
year	.012952	.0010914	11.87	0.000	.0108129 .0150911
_cons	-27.51867	2.153194	-12.78	0.000	-31.73885 -23.29848

Random-effects Parameters	Estimate	Std. Err.	[95% Conf. Interval]
placecode: Identity			
sd(logpop)	.0989279	.0132542	.0760809 .1286358
Residual: AR(1)			
rho	.7900077	.0394951	.6990888 .8557876
sd(e)	.0857877	.0076267	.0720695 .102117

LR test vs. linear regression: chi2(2) = 1506.36 Prob > chi2 = 0.0000

Our model is

$$\begin{aligned}\log_{10}(kwhres_{it}) = & -27.52 + .7086\log_{10}(pop_{it}) - .0011rateres09_{it} - .0039fsumtemp_{it} \\ & + .0003fsumprec_{it} + .0130t + \mu_i\log_{10}(pop_{it}) + .7900\epsilon_{i,t-1} + u_{it}\end{aligned}\quad [13.8]$$

Electricity use in these Arctic towns and villages is predicted by population, price and summer temperature. Unlike lower latitudes where warm summers often mean more electricity use for air conditioning, in these Arctic places warmer (and often, less rainy) summers can encourage people to spend more time outdoors. Adjusting for the effects of population, price and weather, we see an overall upward trend in use as life became more electricity-intensive. Finally, significant variation in the random slope on population indicates that the per-person effects differ from place to place. They tend to be largest in the most northerly region, communities of the North Slope Borough. The autoregressive AR(1) error term proves statistically significant as well.

As noted in Chapter 12, time series often are tested for stationarity before modeling, to avoid reaching spurious conclusions. Differencing provides one set of tools. Alternatively, even when the original series are nonstationary, as in this example (or the ARMAX models in Chapter 12), we can seek a linear combination of the variables that is stationary (*cointegration*; see Hamilton 1994). Model [13.8] serves well for this purpose. None of the 42 residual series (one series for each community) generated by this model show significant autocorrelation, as tested by Ljung-Box portmanteau Q statistics. Thus, residuals are not distinguishable from white noise, a covariance stationary process. The following commands calculate predicted values taking the autoregressive term into account (*yhat_xt*), then use those predictions to calculate residuals (*resid_xt*). White-noise Q tests (**wntestq**) showing no residual autocorrelation are illustrated below for the first 3 out of 42 communities.

```
. predict yhat_xt, fitted
. generate resid_xt = logkwhsold - yhat_xt
. replace yhat_xt = yhat_xt + (.7900077*L1.resid_xt)
. gen yhat_xt10 = 10^yhat_xt
. replace resid_xt = logkwhsold - yhat_xt
. label variable yhat_xt "predicted values log(million kWh)"
. label variable yhat_xt10 "predicted values in millions of kWh"
. label variable resid_xt "residuals log(million kWh)"
. wntestq resid_xt if place == "Ambler city", lags(5)

Portmanteau test for white noise
_____
Portmanteau (Q) statistic =      4.3048
Prob > chi2(5)              =      0.5064

. wntestq resid_xt if place == "Anaktuvuk Pass city", lags(5)

Portmanteau test for white noise
_____
Portmanteau (Q) statistic =      2.3503
Prob > chi2(5)              =      0.7989

. wntestq resid_xt if place == "Aniak city", lags(5)

Portmanteau test for white noise
_____
Portmanteau (Q) statistic =      5.6826
Prob > chi2(5)              =      0.3383
```

Similar tests for all 42 communities found that none of the residual series exhibited significant autocorrelation.

Mixed-Effects Logit Regression

Since 1972, the General Social Survey (Davis et al. 2005) has tracked U.S. public opinion through a series of annual or biannual surveys, and made the data available for teaching and research. Dataset *GSS_2010_SwS* contains a small subset of variables and observations from the 2010 survey, including background variables along with answers to questions about voting, marijuana, gun control, climate change and evolution. The General Social Survey website provides detailed information about this rich data resource (<http://www3.norc.org/GSS+Website/>).

Contains data from C:\data\GSS_2010_SwS.dta, clear					
. describe					
Contains data from C:\data\GSS_2010_SwS.dta					
obs:	809	General Social Survey			
vars:	19	2010--evolution etc.			
size:	21,843	1 Jul 2012 16:06			
variable	name	storage	display	value	
		type	format	label	
id		int	%8.0g	Respondent ID number	
year		int	%8.0g	GSS year	
wtssall		float	%9.0g	LABCM	
cendiv		byte	%15.0g	probability weight	
logsize		float	%9.0g	cendiv	
age		byte	%8.0g	Census division	
nonwhite		byte	%9.0g	log10(size place in 1,000s, +1)	
sex		byte	%8.0g	age	
educ		byte	%8.0g	Age in years	
married		byte	%9.0g	nonwhite	
income06		byte	%15.0g	Consider self white/nonwhite	
polviews		byte	%12.0g	sex	
bush		byte	%9.0g	Responsible gender	
obama		byte	%9.0g	educ	
postlife		byte	%8.0g	Highest year of schooling	
grass		byte	%9.0g	married	
gunlaw		byte	%9.0g	income	
sealevel		byte	%10.0g	Currently married	
evolve		byte	%9.0g	polviews	
				Total family income	
				Polit views liberal-conservative	
				Voted for Bush in 2004	
				Voted for Obama in 2004	
				Believe in life after death	
				Should marijuana be legalized?	
				Oppose permit required to buy gun	
				Bothered if sea level rose 20 ft	
				Humans developed/ earlier species	

Sorted by: id

The GSS question about evolution will be our focus in this section. This question, one item in a science-literacy quiz, asks whether it is true or false that

Human beings, as we know them today, developed from earlier species of animals.

This question invokes individual beliefs as well as science knowledge. About 55% of the respondents said the statement is true.

. tab evolve

Humans developed/ earlier species	Freq.	Percent	Cum.
False	360	44.50	44.50
True	449	55.50	100.00
Total	809	100.00	

Non-missing values for *evolve* were the criterion for selecting this subset of 809 GSS respondents. “False” responses to *evolve* have been recoded as 0 and “True” as 1. The question of survey weighting, a complicated issue with multilevel modeling and not yet resolved for Stata’s mixed-effects logit command **xtmelogit**, is not addressed here.

Research commonly finds that science literacy increases with education, and also is related to other background factors. In the case of *evolve*, we might expect some connection with political outlook as well. A simple logit regression confirms these hypotheses, finding that male, better educated, and politically moderate to liberal respondents more often believe in human evolution.

```
. logit evolve sex age educ polviews, nolog
```

```
Logistic regression
Number of obs      =      785
LR chi2(4)        =     98.93
Prob > chi2       =    0.0000
Pseudo R2         =    0.0918
Log likelihood = -489.36806
```

evolve	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
sex	-.6089296	.1565972	-3.89	0.000	-.9158545 -.3020047
age	-.008189	.0045313	-1.81	0.071	-.0170701 .0006922
educ	.0990929	.0254359	3.90	0.000	.0492395 .1489463
polviews	-.4482161	.0575731	-7.79	0.000	-.5610573 -.3353749
_cons	1.457699	.4891102	2.98	0.003	.4990611 2.416338

Apart from such individual-level predictors of evolution beliefs, there might be a region-level component as well. Controversies about teaching evolution in schools have been most prominent in the South. Consistent with this impression, a chi-squared test of the GSS data finds significant differences between U.S. census divisions. Acceptance of evolution is highest (89%) among respondents in New England — the states of Connecticut, Maine, Massachusetts, New Hampshire, Rhode Island and Vermont. It is lowest (36%) among those in the E South Central division — the states of Alabama, Kentucky, Mississippi and Tennessee. The Pacific and W South Central divisions are second highest and second lowest, respectively.

```
. tab cendiv evolve, row nof chi2
```

Census division	Humans developed/ earlier species		Total
	False	True	
New England	11.11	88.89	100.00
Middle Atlantic	39.00	61.00	100.00
E North Central	43.88	56.12	100.00
W North Central	42.00	58.00	100.00
South Atlantic	50.81	49.19	100.00
E South Central	64.29	35.71	100.00
W South Central	62.34	37.66	100.00
Mountain	43.55	56.45	100.00
Pacific	27.43	72.57	100.00
Total	44.50	55.50	100.00

Pearson chi2(8) = 48.6890 Pr = 0.000

We could add census division to our regression as a set of indicator variables for divisions 2 through 9, each contrasted with division 1, New England.

```
. logit evolve sex age educ polviews i.cendiv, nolog
```

Logistic regression	Number of obs	=	785
	LR chi2(12)	=	124.92
	Prob > chi2	=	0.0000
Log likelihood = -476.37206	Pseudo R2	=	0.1159

evolve	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
sex	-.5609946	.160387	-3.50	0.000	-.8753473 -.2466419
age	-.0092908	.0046327	-2.01	0.045	-.0183706 -.0002109
educ	.0842967	.0261043	3.23	0.001	.0331333 .1354601
polviews	-.416007	.0591817	-7.03	0.000	-.532001 -.3000131
cendiv					
2	-1.501592	.6612973	-2.27	0.023	-2.797711 -.2054736
3	-1.602085	.6504787	-2.46	0.014	-2.877 -.3271704
4	-1.505793	.6931599	-2.17	0.030	-2.864361 -.1472243
5	-1.843963	.6442829	-2.86	0.004	-3.106734 -.5811918
6	-2.149803	.6973044	-3.08	0.002	-3.516495 -.7831115
7	-2.239585	.6743959	-3.32	0.001	-3.561376 -.9177932
8	-1.454279	.6854426	-2.12	0.034	-2.797722 -.1108363
9	-1.141026	.6642829	-1.72	0.086	-2.442996 .1609447
_cons	3.179554	.8138406	3.91	0.000	1.584455 4.774652

Coefficients on the indicator variables give the shift in y -intercept for that division in comparison with New England. All of these coefficients are negative because the log odds of accepting evolution are lower in other divisions than in New England. As expected, the difference is greatest for division 6, E South Central. Only division 9, Pacific, is not significantly different from New England. Net of these region-level effects, all of the individual-level predictors also show significant effects in the expected directions.

An indicator-variable approach works here because we have only 9 clusters (census divisions), and are testing simple hypotheses about shifts in y -intercepts. With more clusters or more complicated hypotheses, a mixed-effects approach could be more practical. For example, we might include random intercepts for each census division in a mixed-effects logit regression model, as follows. The syntax of **xtmelogit** resembles that of **xtmixed**.

```
. xtmelogit evolve sex age educ polviews || cendiv: , nolog
Mixed-effects logistic regression
Group variable: cendiv
Number of obs = 785
Number of groups = 9
Obs per group: min = 27
avg = 87.2
max = 181
Integration points = 7
Log Likelihood = -487.10546
Wald chi2(4) = 72.41
Prob > chi2 = 0.0000



| evolve   | Coef.     | Std. Err. | z     | P> z  | [95% Conf. Interval] |
|----------|-----------|-----------|-------|-------|----------------------|
| sex      | -.5794058 | .1591076  | -3.64 | 0.000 | -.8912511 -.2675606  |
| age      | -.0086106 | .0045962  | -1.87 | 0.061 | -.0176191 .0003979   |
| educ     | .0910441  | .0259804  | 3.50  | 0.000 | .0401235 .1419647    |
| polviews | -.4300722 | .0588037  | -7.31 | 0.000 | -.5453254 -.3148191  |
| _cons    | 1.541323  | .5135581  | 3.00  | 0.003 | .5347681 2.547879    |


| Random-effects Parameters  | Estimate | Std. Err. | [95% Conf. Interval] |
|----------------------------|----------|-----------|----------------------|
| cendiv: Identity sd(_cons) | .3375877 | .1559346  | .1365241 .8347642    |



LR test vs. logistic regression: chibar2(01) = 4.53 Prob>=chibar2 = 0.0167


```

Random intercepts in the output above exhibit significant variation, judging by a likelihood-ratio test versus an ordinary logistic regression ($p = .0167$), or by the standard deviation of random intercepts (.3376) being more than twice its standard error (.1559). The next commands estimate values for these random intercepts through **predict**, then make a table showing them by *cendiv*. Consistent with the earlier chi-squared and indicator-variable analyses, we see positive random y -intercepts (shifting the total effect up) for the New England and Pacific divisions, but negative random y -intercepts (shifting the total effect down) for E South Central and W South Central divisions.

```
. predict recendiv, reffects
. label variable recendiv "random-effect intercept cendiv"
. table cendiv, contents(mean recendiv)
```

Census division	mean(recendiv)
New England	.464954
Middle Atlantic	.0523787
E North Central	-.0165851
W North Central	.0429461
South Atlantic	-.2134227
E South Central	-.3085578
W South Central	-.4224426
Mountain	.083739
Pacific	.3052154

So by any method, we see a robust pattern of regional differences in evolution beliefs, even controlling for individual factors. Mixed-effects modeling allows us to go farther by testing more detailed ideas about regional differences. The literature identifies education as a key influence on evolution and other science beliefs. Do the effects of education differ by census division? We could test this by including both random intercepts and slopes:

```
. xtmelogit evolve sex age educ polviews || cendiv: educ, nolog

Mixed-effects logistic regression
Number of obs = 785
Group variable: cendiv Number of groups = 9
Obs per group: min = 27
avg = 87.2
max = 181

Integration points = 7
Wald chi2(4) = 71.63
Log likelihood = -486.57368 Prob > chi2 = 0.0000
```

evolve	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
sex	-.5692676	.1595327	-3.57	0.000	-.8819459 -.2565893
age	-.0090823	.0046088	-1.97	0.049	-.0181153 -.0000492
educ	.0924205	.027522	3.36	0.001	.0384784 .1463627
polviews	-.4290164	.0588185	-7.29	0.000	-.5442984 -.3137343
_cons	1.532699	.4979934	3.08	0.002	.5566498 2.508748

Random-effects Parameters	Estimate	Std. Err.	[95% Conf. Interval]
cendiv: Independent			
sd(educ)	.0268374	.011591	.0115108 .0625711
sd(_cons)	5.09e-08	.4784982	0 .

LR test vs. logistic regression: chi2(2) = 5.59 Prob > chi2 = 0.0612

Note: LR test is conservative and provided only for reference.

The standard deviation of random slopes on education is more than twice its standard error, suggesting significant regional variation. The standard deviation of random intercepts, on the other hand, is near zero — meaning there is no place-to-place variation in this term. A simpler model omitting the random intercepts through a **nocons** option gives an identical log likelihood.

```
. xtmelogit evolve sex age educ polviews || cendiv: educ, nolog nocons

Mixed-effects logistic regression
Number of obs = 785
Group variable: cendiv Number of groups = 9
Obs per group: min = 27
avg = 87.2
max = 181

Integration points = 7
Wald chi2(4) = 71.63
Log likelihood = -486.57368 Prob > chi2 = 0.0000
```

evolve	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]
sex	-.5692676	.1595327	-3.57	0.000	-.8819459 -.2565893
age	-.0090823	.0046088	-1.97	0.049	-.0181153 -.0000492
educ	.0924205	.027522	3.36	0.001	.0384784 .1463627
polviews	-.4290164	.0588184	-7.29	0.000	-.5442984 -.3137343
_cons	1.532699	.4979933	3.08	0.002	.55665 2.508748

Random-effects Parameters	Estimate	Std. Err.	[95% Conf. Interval]
cendiv: Identity			
sd(educ)	.0268374	.011591	.0115108 .062571

LR test vs. logistic regression: chibar2(01) = 5.59 Prob>=chibar2 = 0.0090

The simplified random-slope model above implies that education has different effects on evolution beliefs in different parts of the country. To see what those effects are, we can **predict** the random slopes, creating a new variable named *ranelduc*. Total effects of *educ* equal these random effects plus the fixed effect given by the coefficient *_b[educ]*. A constant “variable” named *fixedduc* is generated to display the fixed effects in a table, and a new variable *toteduc* gives the total effects of education, or the slope on *educ* within each census division.

```
. predict ranelduc, reffects
. label variable ranelduc "random-effect slope educ"
. gen toteduc = ranelduc + _b[educ]
. label variable toteduc "total random + fixed-effect slope educ"
. gen fixedduc = _b[educ]
. label variable fixedduc "fixed-effect slope educ (constant)"
. table cendiv, contents(mean fixedduc mean ranelduc mean toteduc)
```

Census division	mean(fixedduc)	mean(ranelduc)	mean(toteduc)
New England	.0924205	.0389457	.1313663
Middle Atlantic	.0924205	.0089432	.1013638
E North Central	.0924205	-.0036121	.0888085
W North Central	.0924205	.0035191	.0959396
South Atlantic	.0924205	-.0148976	.077523
E South Central	.0924205	-.0239878	.0684328
W South Central	.0924205	-.0366144	.0558061
Mountain	.0924205	.0033255	.095746
Pacific	.0924205	.0227141	.1151346

From the table, we can confirm that the total education effects equal fixed plus random effects. Figure 13.10 visualizes these total effects.

```
. graph hbar (mean) toteduc, over(cendiv)
ytitle("Effect of education on log odds believe evolution")
```

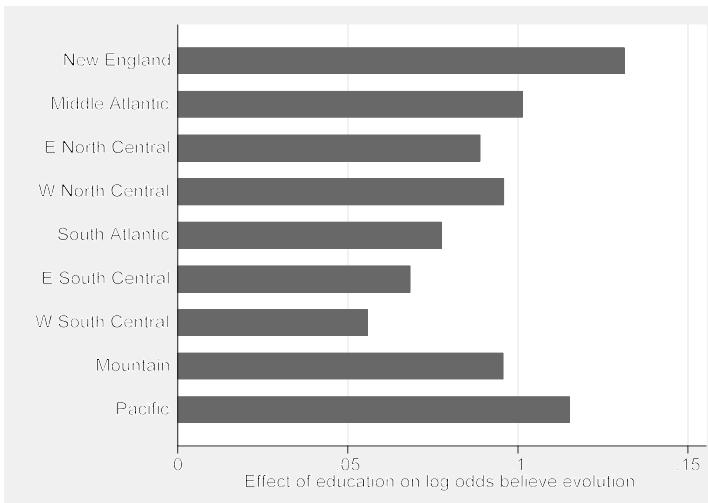


Figure 13.10

We see that education has a positive effect on the log odds of accepting evolution across all census divisions. Education effects are much stronger, however, among respondents in New England or Pacific states compared with the E or W South Central.

Introduction to Programming

As shown earlier, we can create a simple type of program by writing any sequence of Stata commands in a text (ASCII) file. Stata's Do-file Editor (click on Window > Do-file Editor or the icon ) provides a convenient way to do this. After saving the do-file, we enter Stata and type a command with the form **do filename** that tells Stata to read *filename.do* and execute whatever commands it contains. More sophisticated programs are possible as well, making use of Stata's built-in programming language. Most of the commands used in previous chapters actually involve programs written in Stata. These programs might have originated either from StataCorp or from users who wanted something beyond Stata's built-in features to accomplish a particular task.

Stata programs can access all the existing features of Stata, call other programs that call other programs in turn, and use model-fitting aids including matrix algebra and maximum likelihood estimation. Whether our purposes are broad, such as adding new statistical techniques, or narrowly specialized, such as managing a particular database, the ability to write programs in Stata greatly extends what we can do.

Programming is a deep topic in Stata. This brief chapter introduces just a few of the basic concepts and tools, with some examples of how they can be used to facilitate common data-analysis tasks. If you are interested in learning more, Stata's expertly-taught online NetCourses (www.stata.com/netcourse) are a good place to start. The main reference sources are the *Programming Reference Manual* and the two-volume *Mata Matrix Programming Manual*. Details about maximum-likelihood estimation and programming are presented in *Maximum Likelihood Estimation with Stata* (Gould, Pitblado and Poi 2010).

Basic Concepts and Tools

Some elementary concepts and tools, combined with the Stata capabilities described in earlier chapters, suffice to get started.

Do-files

Do-files are text (ASCII) files, created by Stata's Do-file Editor, a word processor, or any other text editor. They are typically saved with a .do extension. The file can contain any sequence of legitimate Stata commands. In Stata, typing the following command causes Stata to read *filename.do* and execute the commands it contains:

```
. do filename
```

Each command in *filename.do*, including the last, must end with a hard return, unless we have reset the delimiter through a **#delimit** command:

```
#delimit ;
```

This sets a semicolon as the end-of-line delimiter, so that Stata does not consider a line finished until it encounters a semicolon. Setting the semicolon as delimiter permits a single command to extend over more than one physical line. Later, we can reset “carriage return” as the usual end-of-line delimiter with another **#delimit** command:

```
#delimit cr
```

A typographical note: Many commands illustrated in this chapter are most likely to be used inside a do-file or ado-file, instead of being typed as a stand-alone command in the Command window. I have written such within-program commands without showing a preceding “.” prompt, as with the two **#delimit** examples above (but not with the **do filename** command, which would have been typed in the Command window as usual).

Ado-files

Ado (automatic do) files are ASCII files containing sequences of Stata commands, much like do-files. The difference is that we need not type **do filename** in order to run an ado-file. Suppose we type the command

```
. clear
```

As with any command, Stata reads this and checks whether an intrinsic command by this name exists. If a **clear** command does not exist as part of the base Stata executable (and, in fact, it does not), then Stata next searches in its usual “ado” directories, trying to find a file named *clear.ado*. If Stata finds such a file (as it should), it then executes whatever commands the file contains.

Ado-files have the extension .ado. User-written programs (written by you) commonly go in a directory named C:\ado\personal, and programs written by other Stata users typically go in C:\ado\plus. The hundreds of official Stata ado-files get installed in C:\Program Files\Stata\ado. Type **sysdir** to see a list of the directories Stata currently uses. Type **help sysdir** or **help adopath** for advice on changing them.

The **which** command reveals whether a given command really is an intrinsic, hardcoded Stata command or one defined by an ado-file; and if it is an ado-file, where that resides. For example, **summarize** is a built-in command, but the **regress** command currently is defined by an ado-file named *regress.ado*, updated in April 2011.

```
. which summarize
built-in command: summarize

. which regress
C:\Program Files\Stata12\ado\base\r\regress.ado
*! version 1.3.0 14apr2011
```

This distinction makes no difference to most users, because **summarize** and **regress** work with similar ease when called. Studying examples and borrowing code from Stata's thousands of official ado-files can be helpful as you get started writing your own. The **which** output above gave the location for file **regress.ado**. To see its actual code, type

```
. viewsource regress.ado
```

The ado-files defining Stata estimation commands have grown noticeably more complicated-looking over the years, as they accommodate new capabilities such as **svy:** prefixes.

Programs

Both do-files and ado-files might be viewed as types of programs, but Stata uses the word “program” in a narrower sense, to mean a sequence of commands stored in memory and executed by typing a particular program name. Do-files, ado-files or commands typed interactively can define such programs. The definition begins with a statement that names the program. For example, to create a program named *count5*, we start with

```
program count5
```

Next should be the lines that actually define the program. Finally, we give an **end** command, followed by a hard return:

```
end
```

Once Stata has read the program definition commands, it retains a definition of the program in memory and will run it any time we type the program's name as a command:

```
. count5
```

Programs effectively make new commands available within Stata, so most users do not need to know whether a given command comes from Stata itself or from an ado-file-defined program.

As we start to write a new program, we often create preliminary versions that are incomplete or unsuccessful. The **program drop** command provides essential help here, allowing us to clear programs from memory so that we can define a new version. For example, to clear program *count5* from memory, type

```
. program drop count5
```

To clear all programs (but not the data) from memory, type

```
. program drop _all
```

Local macros

Macros are names (up to 31 characters) that can stand for strings, program-defined numerical results or user-defined values. A local macro exists only within the program that defines it, and cannot be referred to in another program. To create a local macro named *iterate*, standing for the number 0, type

```
local iterate 0
```

To refer to the contents of a local macro (0 in this example), place the macro name within left and right single quotes. For example,

```
0 display `iterate'
```

Thus, to increase the value of *iterate* by one, we write

```
1 local iterate = `iterate' + 1
      display `iterate'
```

Instead of a number, the macro's contents could be a string or list of words, such as

```
local islands Iceland Faroes
```

To see the string contents, place double quotes around the single-quoted macro name:

```
display " `islands' "
Iceland Faroes
```

We can concatenate further words or numbers, adding to the macro's contents. For example,

```
local islands `islands' Newfoundland Nantucket
      display " `islands' "
Iceland Faroes Newfoundland Nantucket
```

Type **help extended fcn** for information about Stata's "extended macro functions," which extract information from the contents of macros. For instance, we could obtain a count of words in the macro, and store this count as a new macro named *howmany*:

```
4 local howmany: word count `islands'
      display `howmany'
```

Many other extended macro functions exist, with applications to programming.

Global macros

Global macros are similar to local macros, but once defined, they remain in memory and can be used by other programs for the duration of your current Stata session. To refer to a global macro's contents, we preface the macro name with a dollar sign (instead of enclosing the name in left and right single quotes as done with local macros):

```
146      global distance = 73
           display $distance * 2
```

Unless we specifically want to keep macro contents for re-use later in our session, it is better (less confusing, faster to execute, and potentially less hazardous) if we use local rather than global macros in writing programs. To drop a macro from memory, issue a **macro drop** command.

```
macro drop distance
```

We could also drop all macros from memory.

```
macro drop _all
```

Scalars

Scalars can be individual numbers or strings, referenced by a name much as local macros are. To retrieve the contents, however, we do not need to enclose the scalar name in quotes. For example,

```
scalar onethird = 1/3
display onethird
.33333333
display onethird*6
2
```

Scalars are most useful in storing numerical results from calculations, at full numerical precision. Many Stata analytical procedures retain results such as degrees of freedom, test statistics, log likelihoods, and so forth as scalars — as can be seen by typing **return list** or **ereturn list** after the analysis. The scalars, local macros, matrices and functions automatically stored by Stata programs supply building blocks that could be used within new programs.

Version

Stata's capabilities have changed over the years. Consequently, programs written for an older version of Stata might not run directly under the current version. The **version** command works around this problem so that old programs remain usable. Once we tell Stata for what version the program was written, Stata makes the necessary adjustments and the old program can run under

a new version of Stata. For example, if we begin our program with the following statement, Stata interprets all the program's commands as it would have in Stata 9:

```
version 9
```

Typed by itself, the command **version** simply reports the version to which the interpreter is currently set.

Comments

Stata does not attempt to execute any line that begins with an asterisk. Such lines can therefore be used to insert comments and explanations into a program, or interactively during a Stata session. For example,

```
* This entire line is a comment.
```

Alternatively, we can include a comment within an executable line. The simplest way to do so is to place the comment after a double slash, // (with at least one space before the double slash). For example,

```
summarize logsize age // this part is the comment
```

A triple slash (also preceded by at least one space) indicates that what follows, to the end of the line, is a comment; but then the following physical line should be executed as a continuation of the first. For example,

```
summarize logsize age /// this part is the comment  
educ income
```

will be executed as if we had typed

```
summarize logsize age educ income
```

With or without comments, a triple slash tells Stata to read the next line as a continuation of the present line. For example, the following two lines would be read as one **table** command, even though they are separated by a hard return.

```
table married sex, ///  
contents(median age)
```

The triple slash thus provides an alternative to the **#delimit ;** approach described earlier, for writing program commands that are more than one physical line long.

It is also possible to include comments in the middle of a command line, bracketed by /* and */. For example,

```
table married sex, /* this is the comment */ contents(median age)
```

If one line ends with /* and the next begins with */ then Stata skips over the line break and reads both lines as a single command — another line-lengthening trick sometimes found in programs, although /// is now favored.

Looping

There are a number of ways to create program loops. One simple method employs the **forvalues** command. For example, the following program counts from 1 to 5.

```
* Program that counts from one to five
program count5
    version 12.1
    forvalues i = 1/5 {
        display `i'
    }
end
```

By typing these commands, we define program *count5*. Alternatively, we could use the Do-file Editor to save the same series of commands as an ASCII file named *count5.do*. Then, typing the following causes Stata to read the file:

```
. do count5
```

Either way, by defining program *count5* we make this available as a new command:

```
. count5
1
2
3
4
5
```

The command

```
forvalues i = 1/5 {
```

assigns to local macro *i* the consecutive integers from 1 through 5. The command

```
display `i'
```

shows the contents of this macro. The name *i* is arbitrary. A slightly different notation would allow us to count from 0 to 100 by fives (0, 5, 10, ...100):

```
forvalues j = 0(5)100 {
```

The steps between values need not be integers, so long as the endpoints are. To count from 4 to 5 by increments of .01 (4.00, 4.01, 4.02, ...5.00), write

```
forvalues k = 4(.01)5 {
```

Any lines containing valid Stata commands, between the opening and closing curly brackets { }, will be executed repeatedly for each of the values specified. Note that apart from optional comments, nothing on that line follows the opening bracket, and the closing bracket requires a line of its own.

The **foreach** command takes a different approach. Instead of specifying a set of consecutive numeric values, we give a list of items for which iteration occurs. These items could be variables, files, strings, or numeric values. Type **help foreach** to see the syntax of this command.

forvalues and **foreach** create loops that repeat for a pre-specified number of times. If we want looping to continue until some other condition is met, the **while** command is useful. A section of program with the following general form will repeatedly execute the commands within curly brackets, so long as *expression* evaluates to “true”:

```
while expression {
    command A
    command B
    . . .
}
command Z
```

As in previous examples, the closing bracket } should be on its own separate line, not at the end of a command line.

When *expression* evaluates to “false,” the looping stops and Stata goes on to execute *command Z*. Parallel to our previous example, here is a simple program that uses a **while** loop to display onscreen the iteration numbers from 1 through 6:

```
* Program that counts from one to six
program count6
    version 12.1
    local iterate = 1
    while `iterate' <= 6 {
        display `iterate'
        local iterate = `iterate' + 1
    }
end
```

A more substantial loop appears in the *multicat.ado* program described later in this chapter. The *Programming Reference Manual* contains more about programming loops.

If...else

The **if** and **else** commands tell a program to do one thing if an expression is true, and something else otherwise. They are set up as follows:

```
if expression {
```

```

command A
command B
. . .
}
else {
    command Z
}

```

For example, the following program segment checks whether the content of local macro *span* is an odd number, and informs the user of the result.

```

if int(`span'/2) != (`span' - 1)/2 {
    display "span is NOT an odd number"
}
else {
    display "span IS an odd number"
}

```

Arguments

Programs define new commands. In some instances (as with the earlier example, **count5**), we intend our command to do exactly the same thing each time it is used. Often, however, we need a command that is modified by arguments such as variable names or options. There are two ways we can tell Stata how to read and understand a command line that includes arguments. The simplest of these is the **args** command.

The following do-file (*listres1.do*) defines a program that performs a two-variable regression, and then lists the observations with the largest absolute residuals. **listres1** exhibits several bad habits, such as dropping variables and leaving new ones in memory, which could have unwanted side effects. It serves to illustrate the use of temporary variables, however.

```

* Perform simple regression and list observations with #
* largest absolute residuals.
* syntax: listres1 Yvariable Xvariable # IDvariable
capture drop program listres1
program listres1, sortpreserve
    version 12.1
    args Yvar Xvar number id
    quietly regress `Yvar' `Xvar'
    capture drop Yhat_
    capture drop Resid_
    capture drop Absres_
    quietly predict Yhat_
    quietly predict Resid_, resid
    quietly gen Absres_ = abs(Resid_)
    gsort -Absres_
    drop Absres_
    list `id' `Yvar' Yhat_ Resid_ in 1/`number'
end

```

The line **args Yvar Xvar number id** tells Stata to assign arguments to four macros. These arguments could be numbers, variable names or other strings separated by spaces. The first argument becomes the contents of a local macro named *Yvar*, the second a local macro named *Xvar*, and so forth. The program then uses the contents of these macros in other commands, such as the regression:

```
quietly regress `Yvar' `Xvar'
```

The program calculates absolute residuals (*Absres*), and then uses the **gsort** command (with minus sign before the variable name) to sort data in high-to-low order, with missing values last:

```
gsort -Absres_
```

The option **sortpreserve** on the command line makes this program “sort-preserving,” ensuring that the order of the observations is the same after the program runs as it was before.

Dataset *Nations2.dta* contains information on 194 countries, including per capita CO₂ emissions (*co2*), per capita gross domestic product (*gdp*) and country name (*country*). We can open this file, and use it to demonstrate our new program. A **do** command runs do-file *listres1.do*, thereby defining the program and new command **listres1**:

```
. use C:\data\Nations2.dta, clear
. do C:\data\listres1
```

Next, we use the newly-defined **listres1** command, followed by its four arguments. The first argument specifies the *y* variable, the second *x*, the third how many observations to list, and the fourth gives the case identifier. In this example, our command asks for a list of observations that have the five largest absolute residuals.

```
. listres1 co2 gdp 5 country
```

	country	co2	yhat_	Resid_
1.	Qatar	210.65	114.4057	96.2443
2.	Bahrain	102.65	45.54433	57.10566
3.	Trinidad/Tobago	89.25	34.18739	55.06261
4.	Kuwait	118.2	67.83949	50.36051
5.	United Arab Emirates	120.85	79.20002	41.64998

In these five oil-exporting nations, per capita CO₂ emissions are much higher than predicted from GDP.

Syntax

The **syntax** command provides a more complicated but also more powerful way to read a command line. The following do-file named *listres2.do* is similar to our previous example, but it uses **syntax** instead of **args**.

```

* Perform simple or multiple regression and list
* observations with # largest absolute residuals.
* listres2 yvar xvarlist [if] [in], number(#) [id(varname)]
capture drop program listres2
program listres2, sortpreserve
version 12.1
syntax varlist(min=1) [if] [in], Number(integer) [Id(varlist)]
marksample touse
quietly regress `varlist' if `touse'
capture drop Yhat_
capture drop Resid_
capture drop Absres_
quietly predict Yhat_ if `touse'
quietly predict Resid_ if `touse', resid
quietly gen Absres_ = abs(Resid_)
gsort -Absres_
drop Absres_
list `id' `1' Yhat_ Resid_ in 1/`number'
end

```

listres2 has the same purpose as the earlier **listres1**: it performs regression, then lists observations with the largest absolute residuals. This newer version contains several enhancements, made possible by the **syntax** command. It is not restricted to two-variable regression, as was **listres1**. **listres2** will work with any number of predictor variables, including none (in which case, predicted values equal the mean of *y*, and residuals are deviations from the mean). **listres2** permits optional **if** and **in** qualifiers. A variable identifying the observations is optional with **listres2**, instead of being required as it was with **listres1**. For example, we could regress CO₂ emissions on Gross Domestic Product and percent urban, while restricting our analysis to nations in region 2, the Americas.

```

. do C:\data\listres2.do
. listres2 co2 gdp urban if region == 2, n(5) i(country)

```

	country	co2	Yhat_	Resid_
1.	Trinidad/Tobago	89.25	47.63852	41.61148
2.	Barbados	16.65	35.0574	-18.40739
3.	Saint Kitts/Nevis	10.05	26.28106	-16.23106
4.	Antigua and Barbuda	18.3	34.44279	-16.1428
5.	Suriname	19.45	5.137903	14.3121

The **syntax** line in this example illustrates some general features of the command:

```

syntax varlist(min=1) [if] [in], Number(integer) [Id(varlist)]

```

The variable list for a **listres2** command is required to contain at least one variable name (**varlist(min=1)**). Square brackets denote optional arguments — in this example, the **if** and **in** qualifiers, and also the **id()** option. Capitalization of initial letters for the options indicates the minimum abbreviation that can be used. Because the **syntax** line in our example specified **Number(integer)** **Id(varlist)**, an actual command could be written:

```
. listres2 co2 gdp, number(6) id(country)
```

or, equivalently,

```
. listres2 co2 gdp, n(6) i(country)
```

The contents of local macro *number* must be an integer, and *id* one or more variable names.

This example also illustrates the **marksample** command, which marks the subsample (as qualified by **if** and **in**) to be used in subsequent analyses.

The syntax of **syntax** is outlined in the *Programming Manual*. Experimentation and studying other programs help in gaining fluency with this command.

Example Program: multicat (Plot Many Categorical Variables)

The preceding sections presented basic ideas and example short programs. In this section, we apply those ideas to a longer program that defines a new statistical procedure named **multicat**.

Survey research produces datasets containing many categorical variables — sometimes 100 or more. Our 2010 General Social Survey excerpt provides a smaller example with just 19 variables, most of which are categorical responses to survey questions.

```
. use C:\data\GSS_2010_SWS.dta, clear
. describe

Contains data from C:\data\GSS_2010_SWS.dta
    obs:          809                               General Social Survey
    vars:          19                               2010--evolution etc.
    size:        21,843                            20 Jun 2012 06:36

      variable   storage     display       value
      name      type       format      label
      id         int        %8.0g
      year       int        %8.0g
      wtssall    float      %9.0g
      cendifv    byte       %15.0g
      logsize    float      %9.0g
      age        byte       %8.0g
      nonwhite   byte       %9.0g
      sex        byte       %8.0g
      educ       byte       %8.0g
      married    byte       %9.0g
      income06   byte       %15.0g
      polviews  byte       %12.0g
      bush       byte       %9.0g
      obama      byte       %9.0g
      postlife   byte       %8.0g
      grass      byte       %9.0g
      gunlaw     byte       %9.0g
      sealevel   byte       %10.0g
      evolve     byte       %9.0g

      variable label
      id         Respondent ID number
      year       GSS year
      wtssall   probability weight
      cendifv   Census division
      logsize   log10(size place in 1,000s, +1)
      age        age
      nonwhite  Consider self white/nonwhite
      sex        Respondent gender
      educ       Highest year of schooling
      married   Currently married
      income06  Total family income
      polviews  Polit views liberal-conservative
      bush       Voted for Bush in 2004
      obama      Voted for Obama in 2004
      postlife   Believe in life after death
      grass      Should marijuana be legalized?
      gunlaw     Oppose permit required to buy gun
      sealevel   Bothered if sea level rose 20 ft
      evolve     Humans developed/ earlier species

Sorted by: id
```

As a first step in exploring such data, or to prepare a preliminary report, we might simply construct tables showing percentage distributions for each variable. The following command would produce eight such tables, for all variables in the dataset from *polviews* to *evolve*.

```
. tab1 polviews - evolve
```

Stata provides no similarly easy way to draw and save bar charts for a variable list, however. As a programming example, this section shows a makeshift program that was written to meet that particular need when it arose during a complex survey project.

Program **multicat**, defined by the ado-file that follows, builds upon another user-written program named **catplot**, introduced in Chapter 4. **catplot** can draw a variety of graphs showing the distribution of a categorical variable. **multicat** is more specialized, producing only horizontal bar charts for percentages in each category; but this is a particularly useful format for presenting survey data. **multicat** adds the ability to handle a list of many variables, which **catplot** and other Stata graphing commands cannot do. Thus, we could ask **multicat** to draw bar charts of all the variables in our dataset, saving each graph individually along the way. As written, the program saves graphs both in Stata (gph) format and in one other format (emf, eps or pdf depending on the operating system), with file names based on the variable names. You could change any of these specifications by editing *multicat.ado*, tailoring the program to your own analytical needs.

```
!* version 2.0 21jun2012
*! L. Hamilton, Statistics with Stata (2012)
* Requires catplot.ado installed. Graphs are saved in default directory.
program define multicat
version 12.1
syntax varlist [if] [in] [aweight fweight iweight] ///
[, MISSING BY(varname) OVER(varname) ]
if "`over'" != "" {
    display as error "over() option not allowed with multicat;"
    display as error "use by() option or try catplot command instead."
    exit 198
}
marksample touse, strok novarlist
if "`weight'" != "" local Weighted_ = "Weighted"
if "`c(os)'==`Windows'" {
    local filetype "emf"
}
else if "`c(os)'==`Unix'" {
    local filetype "eps"
}
else if "`c(os)'==`MacOSX'" {
    local filetype "pdf"
}
else {
    display as error "unknown operating system: `c(os)'"
    exit 799
}
capture {
    if "`by'" != "" {
        foreach var of varlist `varlist' {
            local vlab_: variable label `var'
            catplot hbar `var' [`weight' `exp'] if `touse', ///
            }
    }
}
```

```

        xlabel(bar, format(%3.0f)) ///
        percent(`by') ytitle("`Weighted_ Percent") ///
        `missing' by(`by', title("`Vlab_'", size(medium)))
        graph save -`by'-`var'.gph, replace
        graph export -`by'-`var'.`filetype', replace
    }
}
else {
    foreach var of varlist `varlist' {
        quietly tab `var' if `touse', `missing'
        local Nofobs_ = r(N)
        local Vlab_ : variable label `var'
        catplot hbar `var' [`weight' `exp'] if `touse', ///
            xlabel(bar, format(%3.0f)) ///
            percent ytitle("`Weighted_ Percent, N = `Nofobs_'"') ///
            title("`Vlab_'", size(medium)) `missing' `options'
        graph save Graph -`var'.gph, replace
        graph export -`var'.`filetype', replace
    }
}
error _rc
end

```

Indentation of lines has no effect on the program's execution, but makes it easier for the programmer to read. The heart of **multicat** is its **syntax** statement, and then a **foreach** loop that repeatedly calls **catplot** for each variable in the variable list. Local macros pass information to the **catplot** command that actually draws the graphs. The command allows analytical weights (aweights) which have an effect here similar to that of probability weights (pweights) in a **svy: tab** command. It allows **in** or **if** qualifiers. Optionally we could include **missing** values, and use **by()** but not **over()**.

The **multicat** program was written incrementally, beginning with a do-file named *multicat.do*. Starting with input components such as the syntax statement, the do-file was run to see how things worked before adding other components. Not all of the trial runs produced satisfactory results. Typing the command

```
. set trace on
```

causes Stata to display programs line-by-line as they execute, so we can see exactly where an error occurs. Later, we can turn this feature off by typing

```
. set trace off
```

Preliminary *multicat.do* versions contained the first line, **capture program drop multicat**, to discard the program from memory before defining it again. This was necessary during the writing and debugging stage, when a previous version of our program might have been incomplete or incorrect. Such lines should be deleted when the program is mature, however.

Once we believe our do-file defines a program that we will want to use again, we can create an ado-file. This is accomplished just by saving the file with an .ado extension (*multicat.ado*). The recommended location is in *ado\personal*; you might need to create this directory and

subdirectory if they do not already exist. Other locations are possible, but review the *User’s Manual* section on “Where does Stata look for ado-files?” before proceeding. Once this is done, we can use **multicat** as a regular command within Stata.

The program could be refined further to make it more flexible, elegant and user-friendly. Note the inclusion of comments stating the source and “version 2.0” in the first two lines, which both begin `*!` The comment refers to version 2.0 of *multicat.ado*, not Stata (an earlier version of *multicat.ado* appeared in the previous edition of this book). The Stata version suitable for running this program is specified by the **version 12.1** command a few lines later. Although the `*!` comments do not affect how the program runs, they are visible to a **which** command:

```
. which multicat
.\multicat.ado
*! version 2.0 21jun2012
*! L. Hamilton, Statistics with Stata (2012)
```

Using multicat

After *multicat.ado* has been saved (for example, in C:\ado\personal), the command **multicat** becomes usable as if it were an ordinary (albeit, unfinished) feature of Stata. Figure 14.1 shows responses regarding political views. Both percentages and the number of observations appear in the plot.

```
. multicat polviews, missing
```

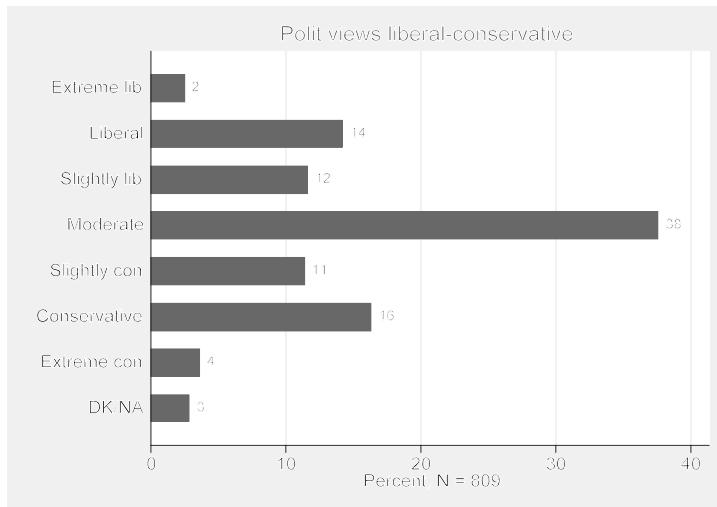


Figure 14.1

Survey data commonly are analyzed using probability weights and **svyset** data, as discussed in Chapter 4. Applying survey weights, **svy: tab** finds the following responses to the question about legalizing marijuana.

```
. svy: tab grass, percent miss
(running tabulate on estimation sample)
```

Number of strata = 1
 Number of PSUs = 809

Number of obs = 809
 Population size = 812.73293
 Design df = 808

Should marijuana be legalized?	percentages
Not Legal	29.68
DK	30.04
NA	5.035
Total	35.24
	100

Key: percentages = cell percentages

This question has two kinds of missing values. About 5% of the respondents were asked the marijuana question but said they did not know, or declined to give an answer. Their missing values have been coded .a, with value label “DK”. Another 35% in this sample were not asked the *grass* question. Those have been coded .b, with value label “NA”. The GSS accommodates a large number of questions by asking different questions to subsets of their sample. Analytically it makes sense for us to set aside the un-asked group and recalculate the percentages. Doing so, we find an even split: about 46% in favor and 46% opposed to legalizing marijuana, with 8% undecided.

```
. svy: tab grass if grass < .b, percent miss
(running tabulate on estimation sample)
```

Number of strata = 1
 Number of PSUs = 524

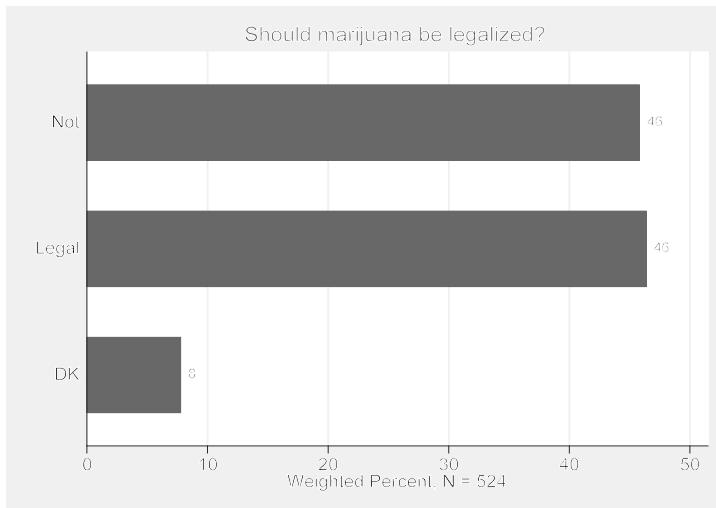
Number of obs = 524
 Population size = 526.30952
 Design df = 523

Should marijuana be legalized?	percentages
Not Legal	45.83
DK	46.39
Total	7.776
	100

Key: percentages = cell percentages

multicat (based on **catplot**) does not understand **svy:** commands or probability weights, but analytical weights (**aweights**) have the same effect here. Figure 14.2 shows a **multicat** plot corresponding to the table above. The **multicat** graphic also notes the relevant sample size (524).

```
. multicat grass if grass < .b [aw = wtssall], miss
```

**Figure 14.2**

A series of many graphs like Figure 14.2, dropped into a document or slide show, could be read and annotated by the analyst for a quick presentation of results. Setting aside the complication of missing values, here is a quick example where **multicat** draws a set of 8 bar charts for the variables from *polviews* to *evolve*. **multicat** automatically saves each graph with filenames such as *-polviews.gph*. A **graph combine** command then fits the graphs into one image, Figure 14.3.

```
. multicat polviews-evolve [aw = wtssall]
.
graph combine -polviews.gph -bush.gph -obama.gph -postlife.gph
               -grass.gph -gunlaw.gph -sealevel.gph -evolve.gph
```

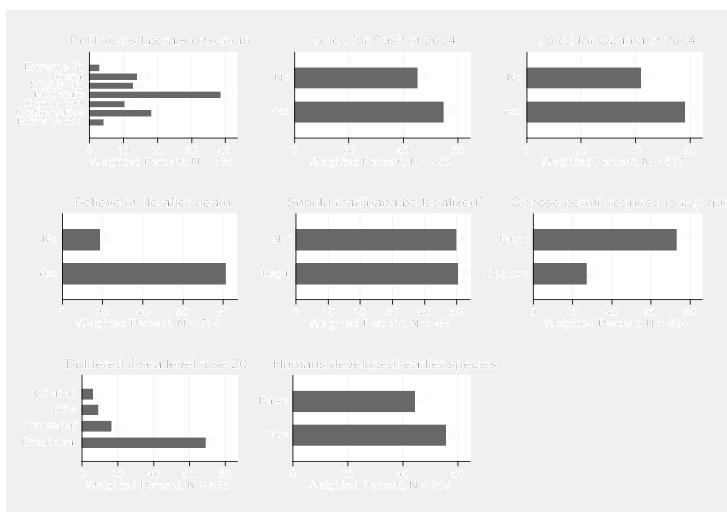


Figure 14.3

Survey research becomes more interesting as we start to compare subgroups. For example, the even split on legalizing marijuana looks different when broken down by gender in Figure 14.4.

```
. multicat grass [aw = wtssall], by(sex)
```

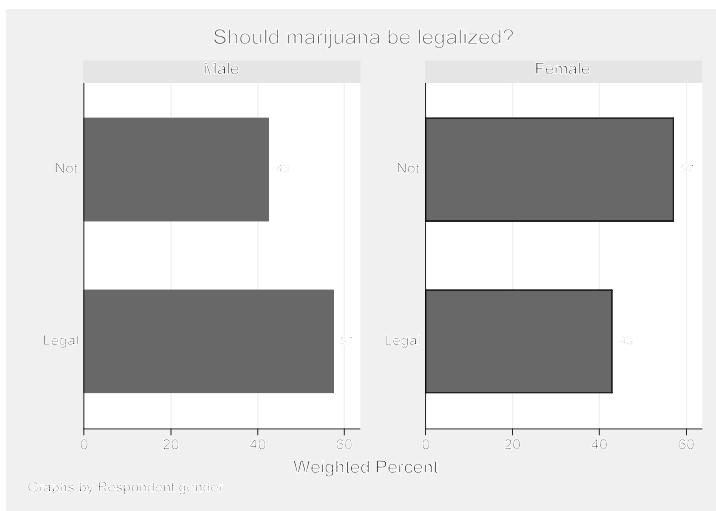


Figure 14.4

This is where **multicat** comes into its own, offering an easy way to draw many comparison charts. The following commands draw 8 bar charts for opinions broken down by gender, then combine 4 of the separate charts into one image, Figure 14.5. We see that female respondents are more likely than males to believe in life after death (86 versus 76%), oppose legalizing

marijuana (57 versus 43%), favor gun permits (77 versus 68%) or reject evolution (49 versus 39%).

```
. multicut polviews-evolve [aw = wtssall], by(sex)
. graph combine -sex-postlife.gph -sex-grass.gph -sex-gunlaw.gph
    -sex-evolve.gph
```

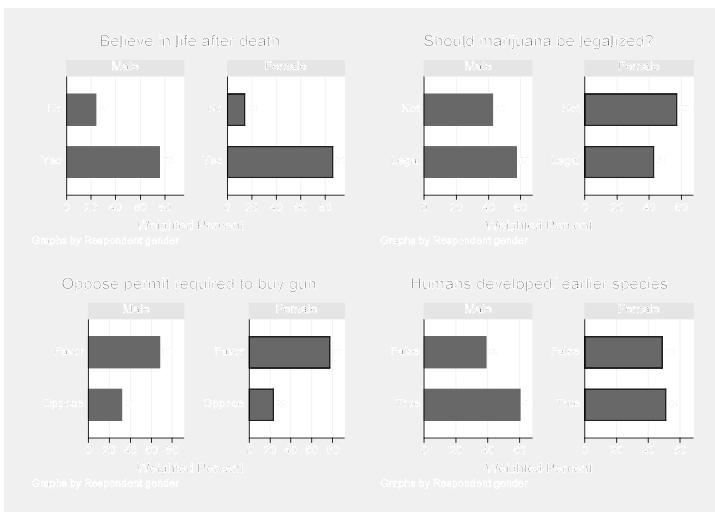


Figure 14.5

The **graph combine** step is used in Figures 14.3 and 14.5 for compact illustrations of what **multicut** is doing. In research, the actual number of graphs often exceeds what we want to crowd into one image. A **multicut** command could just as easily draw 100 graphs comparing survey responses by men and women, then 100 more comparing responses by education level, age group, political party, region or any other categories of interest. Most analysts will never need this particular trick, but when specialized needs arise in a project, makeshift programs of this sort could become essential.

Help File

Help files are an integral aspect of using Stata. For a user-written program such as *multicut.ado*, they become even more important because no documentation exists in the manuals. We can write a help file for *multicut.ado* by using Stata's Do-file Editor to create a text file named *multicut.sthlp*. This help file should be saved in the same ado-file directory (for example, C:\ado\personal) as *multicut.ado*.

Any text file, saved in one of Stata's recognized ado-file directories with a name of the form *filename.sthlp*, will be displayed onscreen by Stata when we type **help filename**. For example, we might write the following in the Do-file Editor, and save it in directory C:\ado\personal as file *multicut.sthlp*. Typing **help multicut** at any time would then cause Stata to display the text.

```
multicat -- Multiple bar charts of categorical variables}

multicat varlist [aw = weightvar] [if exp] [in range],
[missing] [by(groupvar)]
```

Description

`multicat` draws horizontal bar charts showing percentages of categorical variables. It saves one chart for each of the variables in `varlist`. Graphs are saved in the current default directory, with file names based on variable names preceded by a hyphen, such as `-vote.gph` or `-region-vote.gph`. They are saved both in Stata's `.gph` format and one other graphical file format (`.emf`, `.eps` or `.pdf`) depending on operating system.

Using analytical weights [`aw = weightvar`] with `multicat` will result in percentages equivalent to those obtained by `svy: tab` applied to data declared as survey type, by a command such as `svyset [pw = weightvar]`. The `svy:` prefix cannot be used with `multicat` itself. Chapter 14 in *Statistics with Stata* (2012) has examples and discussion of `multicat`.

`multicat` requires that `catplot` is installed. Type `findit catplot` for instructions on installing this unofficial program, written by Nicholas Cox.

Options

`missing` includes missing values in the bar chart and calculated percentages.

`by(groupvar)` draws an image containing separate small charts for each value of `groupvar`.

Examples

```
multicat party wrongtrack vote

multicat party-vote [aw = weightvar], miss

multicat party-vote [aw = weightvar], by(region)
```

References

Hamilton, Lawrence C. 2012.
Statistics with Stata. Belmont, CA: Cengage

Nicer help files containing links, text formatting, dialog boxes, and other features can be designed using Stata Markup and Control Language (SMCL). All official Stata help files, as well as log files and onscreen results, employ SMCL. A recommended standard outline for help files appears in the *User's Guide*.

The following is an SMCL version of the help file for **multicat**, roughly following the *User's Guide* outline. Once this file has been saved in ado\personal with the file name *multicat.sthlp*, typing **help multicat** will produce a readable and official-looking display.

```
{smcl}
{* *! version 2.0 21jul2012}{...}
{cmd:help multicat}
{hline}

{title:Title}
{phang}
{bf:multicat -- Multiple bar charts of categorical variables}

{title:Syntax}
{p 8 17 12}
{cmd:multicat} {it:varlist} [{it:weight}] [{cmd;if} {it:exp}]
[ {cmd:in} {it:range}] {cmd:,} [ {cmdab:miss:ing}]
[ {cmd:by()} {it:groupvar}{cmd:}]

{title:Description}

{pstd}
{cmd:multicat} draws horizontal bar charts showing percentages of categorical variables. It saves one chart for each of the variables in {it:varlist}. Graphs are saved in the current default directory, with file names based on variable names preceded by a hyphen, such as {it:-vote.gph} or {it:-region-vote.gph}. They are saved both in Stata's .gph format and one other graphical file format (.emf, .eps or .pdf) depending on operating system.

{pstd}
Using analytical weights {cmd:[aw = ]{it:weightvar}{cmd:}} with {cmd:multicat} will result in percentages equivalent to those obtained by {cmd:svy: tab} applied to data declared as survey type, by a command such as {cmd:svyset [pw = ]{it:weightvar}{cmd:}}. The {cmd:svy:} prefix cannot be used with {cmd:multicat} itself. Chapter 1 4 i n { b r o w s e "http://www.stata.com/bookstore/statistics-with-stata/index.html": Statistics with Stata} (2012) has examples and discussion of {cmd:multicat}.

{pstd}{cmd:multicat} requires that {cmd:catplot} is installed. Type {cmd:findit catplot} for instructions on installing this unofficial program, written by Nicholas Cox.

{title:Options}

{phang}
{cmdab:miss:ing} includes missing values in the bar chart and calculated percentages.

{phang}
{cmd:by()} {it:groupvar}{cmd:} draws an image containing separate small charts for each value of {it:groupvar}.

{title:Examples}
```

```

{phang}
{cmd:. multicat party wrongtrack vote}

{phang}
{cmd:. multicat party-vote [aw = weightvar], miss}

{phang}
{cmd:. multicat party-vote [aw = weightvar], by(region)}

{title:References}

{pstd}
Hamilton, Lawrence C. 2012.
{browse "http://www.stata.com/bookstore/statistics-with-stata/index.html": Statistics with Stata}. Belmont, CA: Cengage.{p_end}

```

The help file begins with **{smcl}**, which tells Stata to process the file as SMCL. Curly brackets **{ }** enclose SMCL codes, many of which have the form **{command:text}** or **{command arguments:text}**. The following examples illustrate how these codes are interpreted.

{cmd:help multicat} Display the text “help multicat” as a command. That is, show “help multicat” with whatever colors and font attributes are presently defined as appropriate for a command.

{hline} Draw a horizontal line.

{title>Title} Display the text “Title” as a title.

{phang} Hanging indentation for the following paragraph.

{bf:multicat – ...} Display text in **boldface**.

{p 8 17 12} Format the following text as a paragraph, with the first line indented 8 characters, subsequent lines indented 17, and the right margin brought in 12 characters.

{it:varlist} Display the text *varlist* in italics.

{cmdab:miss:ing} Display “missing” as a command, with the letters “miss” marked as the minimum abbreviation.

{browse "http://www.stata.com/bookstore/statistics-with-stata/index.html":Statistics...}
Link the text “Statistics with Stata” to the web address (URL)
<http://www.stata.com/bookstore/statistics-with-stata/index.html>. Clicking on “Statistics with Stata” should then launch your browser and connect it to this URL.

The *Programming Manual* supplies details about using these and many other SMCL commands.

Monte Carlo Simulation

Monte Carlo simulations generate and analyze many samples of artificial data, allowing researchers to investigate the long-run behavior of their statistical techniques. The **simulate** command makes designing a simulation straightforward so that it only requires a small amount of additional programming. This section gives two examples.

To begin a simulation, we need to define a program that generates one sample of random data, analyzes it, and stores the results of interest in memory. The following file defines an r-class program (one capable of storing r() results) named **meanmedian**. This program randomly generates 100 values of variable *x* from a standard normal distribution. It next generates 100 values of variable *w* from a contaminated normal distribution: N(0,1) with probability .95, and N(0,10) with probability .05. Contaminated normal distributions have often been used in robustness studies to simulate variables that contain occasional wild errors. For both variables, **meanmedian** obtains means and medians.

```
* Creates a sample containing n=100 observations of variables x and w.
* x~N(0,1)                                x is standard normal
* w~N(0,1) with p=.95, w~N(0,10) with p=.05    w is contaminated normal
* Calculates the mean and median of x and w.
* Stored results: r(xmean) r(xmedian) r(wmean) r(wmedian)
program meanmedian, rclass
    version 12.1
    drop _all
    set obs 100
    generate x = rnormal()
    summarize x, detail
    return scalar xmean = r(mean)
    return scalar xmedian = r(p50)
    generate w = rnormal()
    replace w = 10*w if runiform() < .05
    summarize w, detail
    return scalar wmean = r(mean)
    return scalar wmedian = r(p50)
end
```

Because we defined **meanmedian** as an r-class command, like **summarize**, it can store its results in r() scalars. **meanmedian** creates four such scalars: *r(xmean)* and *r(xmedian)* for the mean and median of *x*; *r(wmean)* and *r(wmedian)* for the mean and median of *w*.

Once **meanmedian** has been defined, whether through a do-file, ado-file, or typing commands interactively, we can call this program with a **simulate** command. To create a new dataset containing means and medians of *x* and *w* from 5,000 random samples, type

```
. simulate xmean = r(xmean) xmedian = r(xmedian) wmean = r(wmean)
                wmedian = r(wmedian), reps(5000): meanmedian
```

```
command: meanmedian
xmean: r(xmean)
xmedian: r(xmedian)
wmean: r(wmean)
wmedian: r(wmedian)
```

Simulations (5000)

This command creates new variables *xmean*, *xmedian*, *wmean* and *wmedian*, based on the *r()* results from each iteration of **meanmedian**.

```
. describe
```

Contains data

obs:	5,000
vars:	4
size:	80,000

simulate: meanmedian
20 Jun 2012 13:07

variable	storage type	display format	value label	variable label
xmean	float	%9.0g	r(xmean)	
xmedian	float	%9.0g	r(xmedian)	
wmean	float	%9.0g	r(wmean)	
wmedian	float	%9.0g	r(wmedian)	

Sorted by:

```
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
xmean	5000	-.0000624	.1000338	-.3582268	.3759606
xmedian	5000	.0001874	.1258506	-.4006759	.5206654
wmean	5000	.0014439	.243858	-1.067343	1.012038
wmedian	5000	-.0024416	.1294947	-.496728	.4454642

The means of these means and medians, across 5,000 samples, are all close to 0 — consistent with our expectation that the sample mean and median should both provide unbiased estimates of the true population means (0) for *x* and *w*. Also as theory predicts, the mean exhibits less sample-to-sample variation than the median when applied to a normally distributed variable (*x*). The standard deviation of *xmedian* is .126, noticeably larger than the standard deviation of *xmean* (.100). When applied to the non-normal, outlier-prone variable *w*, on the other hand, the opposite holds true: the standard deviation of *wmedian* is much lower than the standard deviation of *wmean* (.129 versus .244). This Monte Carlo experiment demonstrates that the median remains a relatively stable measure of center despite wild outliers in the contaminated distribution, whereas the mean breaks down and varies much more from sample to sample. Figure 14.6 draws the comparison graphically as box plots (and, incidentally, demonstrates how to control the shapes of box plot outlier-marker symbols). To make room for four variables within the one-row legend, symbols are drawn with only half their usual width (**symxsize(*.5)**).

```
. graph box xmean xmedian wmean wmedian, yline(0)
legend(row(1) symxsize(*.5))
marker(1, msymbol(+)) marker(2, msymbol(Th))
marker(3, msymbol(Oh)) marker(4, msymbol(Sh))
```

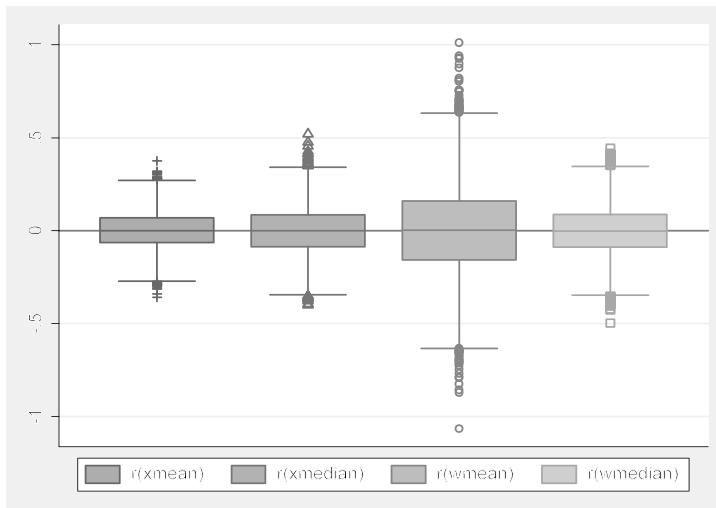


Figure 14.6

Our next example extends the inquiry to robust methods, bringing together several themes from this book. Program **regsim** generates 100 observations of x (standard normal) and two y variables. y_1 is a linear function of x plus standard normal errors. y_2 is also a linear function of x , but adding contaminated normal errors. These variables help to explore how various regression methods behave in the presence of normal and nonnormal, heavy-tailed error distributions. Four methods are employed: ordinary least squares (**regress**), robust regression (**rreg**), quantile regression (**qreg**), and quantile regression with bootstrapped standard errors (**bsqreg**, with 500 repetitions). Robust and quantile regression (Chapter 8) in theory should prove more resistant to the effects of outliers. Through this Monte Carlo experiment, we test whether that is true. **regsim** applies each method to the regression of y_1 on x and then to the regression of y_2 on x . For this exercise, the program is defined by an ado-file, *regsim.ado*, saved in the ado\personal directory.

```
program regsim, rclass
* Performs one iteration of a Monte Carlo simulation comparing
* OLS regression (regress) with robust (rreg) and quantile
* (qreg and bsqreg) regression. Generates one n = 100 sample
* with x ~ N(0,1) and y variables defined by the models:
*
* MODEL 1: y1 = 2x + e1    e1 ~ N(0,1)
*
* MODEL 2: y2 = 2x + e2    e2 ~ N(0,1) with p = .95
*           e2 ~ N(0,10) with p = .05
*
* Bootstrap standard errors for qreg involve 500 repetitions.
*
        version 12.1
        if "`1'" == "?" {
            global S_1 "b1 b1r se1r b1q se1q se1qb ///
                        b2 b2r se2r b2q se2q se2qb"
            exit
        }

```

```

drop _all
set obs 100
generate x = rnormal()
generate e = rnormal()
generate y1 = 2*x + e
reg y1 x
    return scalar B1 = _b[x]
rreg y1 x, iterate(25)
    return scalar B1R = _b[x]
    return scalar SE1R = _se[x]
qreg y1 x
    return scalar B1Q = _b[x]
    return scalar SE1Q = _se[x]
bsqreg y1 x, reps(500)
    return scalar SE1QB = _se[x]
replace e = 10 * e if runiform() < .05
generate y2 = 2*x + e
reg y2 x
    return scalar B2 = _b[x]
rreg y2 x, iterate(25)
    return scalar B2R = _b[x]
    return scalar SE2R = _se[x]
qreg y2 x
    return scalar B2Q = _b[x]
    return scalar SE2Q = _se[x]
bsqreg y2 x, reps(500)
    return scalar SE2QB = _se[x]
end

```

The r-class program stores coefficient or standard error estimates from eight regression analyses. These results are given names such as

- $r(B1)$ coefficient from OLS regression of $y1$ on x
- $r(B1R)$ coefficient from robust regression of $y1$ on x
- $r(SE1R)$ standard error of robust coefficient from model 1

and so forth. All the robust and quantile regressions involve multiple iterations: typically five to ten iterations for **rreg**, about five for **qreg**, and several thousand for **bsqreg** with its 500 bootstrap re-estimations of about five iterations each, *per sample*. Thus, a single execution of **regsim** demands more than 2,000 regressions. The following command calls for five repetitions, requiring more than 10,000 regressions.

```

. simulate b1 = r(B1) b1r = r(B1R) selr = r(SE1R)
    b1q = r(B1Q) selq = r(SE1Q) selqb = r(SE1QB) b2 = r(B2)
    b2r = r(B2R) se2r = r(SE2R) b2q = r(B2Q) se2q = r(SE2Q)
    se2qb = r(SE2QB), reps(5): regsim

```

You might want to run a small simulation like this as a trial to get a sense of the time required on your computer. For research purposes, however, we would need a larger experiment. Dataset *regsim.dta* contains results from an experiment involving 500 repetitions of **regsim** — more than a million regressions. The regression coefficients and standard error estimates produced by this experiment are summarized below.

```

. describe

```

Contains data from C:\data\regsim.dta			Monte Carlo estimates of b in 500 samples of n=100 21 Jun 2012 08:17		
obs:	500	vars:	12	size:	24,000
variable	name	storage type	display format	value label	variable label
b1		float	%9.0g	r(B1)	
b1r		float	%9.0g	r(B1R)	
se1r		float	%9.0g	r(SE1R)	
b1q		float	%9.0g	r(B1Q)	
se1q		float	%9.0g	r(SE1Q)	
se1qb		float	%9.0g	r(SE1QB)	
b2		float	%9.0g	r(B2)	
b2r		float	%9.0g	r(B2R)	
se2r		float	%9.0g	r(SE2R)	
b2q		float	%9.0g	r(B2Q)	
se2q		float	%9.0g	r(SE2Q)	
se2qb		float	%9.0g	r(SE2QB)	

Sorted by:

. summarize

Variable	Obs	Mean	Std. Dev.	Min	Max
b1	500	1.99586	.104467	1.692893	2.293595
b1r	500	1.996901	.1077322	1.698501	2.294482
se1r	500	.1046559	.0108543	.0789753	.1523494
b1q	500	1.99658	.1246462	1.638891	2.370703
se1q	500	.13393	.0206363	.0801532	.2059937
se1qb	500	.1373001	.0321417	.0532386	.2581546
b2	500	1.986367	.2604184	1.066318	2.90484
b2r	500	1.997187	.1127494	1.674992	2.307488
se2r	500	.1087309	.0117741	.081064	.1564037
b2q	500	1.996925	.1314325	1.591606	2.370703
se2q	500	.1416007	.0212944	.0880669	.2220859
se2qb	500	.1456451	.0343871	.0560117	.2704635

Figure 16.8 draws the distributions of coefficients as box plots. To make the legend readable we use the **legend(symxsize(*.3) colgap(*.3))** options, which set the width of symbols and the gaps between columns within the legend at just 30% of their default size. **help legend option** and **help relativesize** supply further information about these options.

```
.graph box b1 b1r b1q b2 b2r b2q, ytitle("Estimates of slope (b=2)")  
yline(2) legend(row(1) symxsize(*.3) colgap(*.3)  
label(1 "OLS 1") label(2 "robust 1") label(3 "quantile 1")  
label(4 "OLS 2") label(5 "robust 2") label(6 "quantile 2"))
```

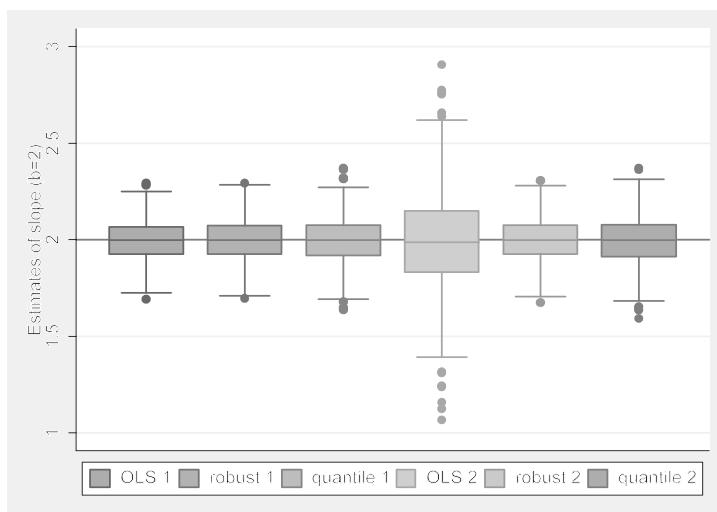


Figure 14.7

All three regression methods (OLS, robust and quantile) produced mean coefficient estimates for both models that are not significantly different from the true value, $\beta = 2$. This can be confirmed through t tests such as

```
. ttest b2r = 2
One-sample t test

```

Variable	Obs	Mean	Std. Err.	Std. Dev.	[95% Conf. Interval]
b2r	500	1.997187	.0050423	.1127494	1.987281 2.007094

```

mean = mean(b2r)
Ho: mean = 2
t = -0.5578
degrees of freedom = 499
Ha: mean < 2
Pr(T < t) = 0.2886
Ha: mean != 2
Pr(|T| > |t|) = 0.5772
Ha: mean > 2
Pr(T > t) = 0.7114

```

All the regression methods thus yield unbiased estimates of β , but they differ in their sample-to-sample variation or efficiency. Applied to the normal-errors model 1, OLS proves the most efficient, as the Gauss–Markov theorem predicts. The observed standard deviation of OLS coefficients is .1045, compared with .1077 for robust regression and .1246 for quantile regression. Relative efficiency, expressing an OLS coefficient's observed variance as a percentage of another estimator's variance, provides a standard way to compare such statistics:

```
. quietly summarize b1
. scalar Varb1 = r(Var)
. quietly summarize b1r
. display 100*(Varb1/r(Var))
94.030265

. quietly summarize b1q
. display 100*(Varb1/r(Var))
70.242519
```

The calculations above use the `r(Var)` variance result from **summarize**. We first obtain the variance of the OLS estimates b_1 , and make this value scalar $Varb1$. Next the variances of the robust estimates b_{1r} , and the quantile estimates b_{1q} , are obtained and each is compared with $Varb1$. This reveals that robust regression was about 94% as efficient as OLS when applied to the normal-errors model — close to the large-sample efficiency of 95% that this robust method theoretically should have (Hamilton 1992a). Quantile regression, in contrast, achieves a relative efficiency of only 70% with the normal-errors model.

Similar calculations for the contaminated-errors model tell a different story. OLS was the best (most efficient) estimator with normal errors, but with contaminated errors it becomes the worst:

```
. quietly summarize b2
. scalar Varb2 = r(Var)
. quietly summarize b2r
. display 100*(Varb2/r(Var))
533.47627

. quietly summarize b2q
. display 100*(Varb2/r(Var))
392.58875
```

Outliers in the contaminated-errors model cause OLS coefficient estimates to vary wildly from sample to sample, as can be seen in the fourth box plot of Figure 14.7. The variance of these OLS coefficients is more than five times greater than the variance of the corresponding robust coefficients, and almost four times greater than that of quantile coefficients. Put another way, both robust and quantile regression prove to be much more stable than OLS in the presence of outliers, yielding correspondingly lower standard errors and narrower confidence intervals. Robust regression outperforms quantile regression with both the normal-errors and the contaminated-errors models.

Figure 14.8 illustrates the comparison between OLS and robust regression with a scatterplot showing 500 pairs of regression coefficients. The OLS coefficients (vertical axis) vary much more widely around the true value, 2.0, than `rreg` coefficients (horizontal axis) do.

```
. graph twoway scatter b2 b2r, msymbol(dh) ylabel(1(.5)3, grid)
    yline(2) xlabel(1(.5)3, grid gmin gmax) xline(2)
    ytitle("OLS regression coef, contaminated errors")
    xtitle("Robust regression coef, contaminated errors")
```

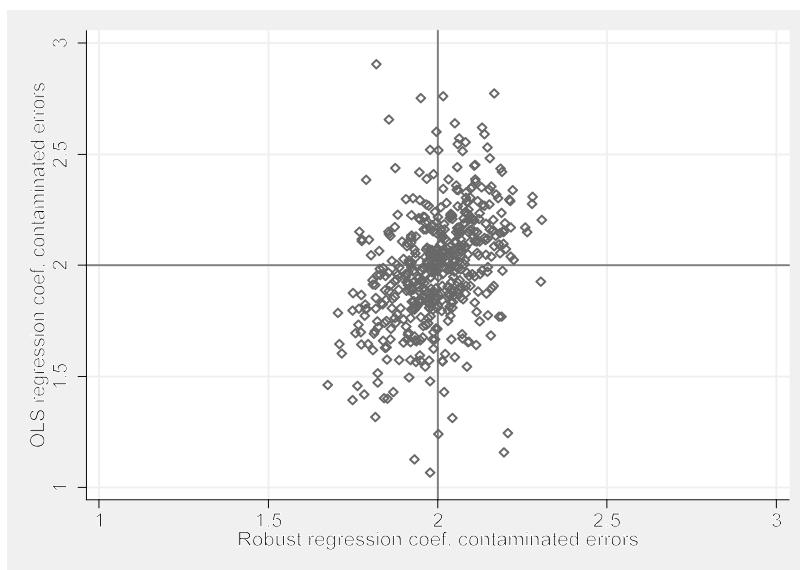


Figure 14.5

The Monte Carlo experiment also provides information about the estimated standard errors under each method and model. Mean estimated standard errors differ from the observed standard deviations of coefficients. Discrepancies for the robust standard errors are relatively small — less than 4%. For the theoretically-derived quantile standard errors the discrepancies are larger, around 7%. The least satisfactory estimates appear to be the bootstrapped quantile standard errors obtained by **b5qreg**. Means of the bootstrap standard errors exceed the observed standard deviation of $b1q$ and $b2q$ by 10 or 11%. Bootstrapping apparently over-estimates the sample-to-sample variation.

Monte Carlo simulation has become a key method in modern statistical research, and it plays a growing role in statistical teaching as well. These examples demonstrate some easy ways to get started.

Matrix Programming with Mata

Stata's matrix programming language called Mata, is described in the two-volume *Mata Matrix Programming* manual. This rich topic lies beyond the introductory scope of *Statistics with Stata*. It seems fitting, however, to conclude the book with a brief look at Mata. Its programming tools open new paths for Stata's development.

Rather than undertaking the large task of explaining Mata's concepts and features, we will proceed inductively and jump right to an example: writing a program that performs ordinary least squares (OLS) regression. The basic regression model is

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{u}$$

where \mathbf{y} is an $(n \times 1)$ column vector of dependent-variable values, \mathbf{X} an $(n \times k)$ matrix containing values of (usually) $k - 1$ predictor variables and a column of 1's, and \mathbf{u} an $(n \times 1)$ vector of errors. \mathbf{b} is a $(k \times 1)$ vector of regression coefficients, estimated as

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

This matrix calculation, familiar to generations of statistics students, provides a good entry point for seeing Mata at work.

Dataset *reactor.dta* contains information about the decommissioning costs of five nuclear power plants that were shut down over 1968–1982. This example has the pedagogical advantage that its small matrices could be written easily on blackboard or paper, if desired (e.g., Hamilton 1992a:340). In any event, it invites the question of how decommissioning costs might be related to reactor capacity and years of operation.

```
. use C:\data\reactor.dta, clear
. describe

Contains data from C:\data\reactor.dta
obs: 5
vars: 6
size: 110

variable   storage      display      value
name       type        format     label
variable label

site        str14    %14s
decom       byte      %8.0g
capacity    int       %8.0g
years       byte      %9.0g
start       int       %8.0g
close       int       %8.0g

Reactor decommissioning costs
(from Brown et al. 1986)
20 Jun 2012 13:23
```

Sorted by: start

Performing OLS regression with Stata is very easy, of course. We find that decommissioning costs among these five reactors increased by about .176 million dollars (\$175,874) with each megawatt of generating capacity, and by about 3.9 million dollars with each year of operation. The two predictors explain almost 99% of the variance in decommissioning costs ($R^2_a = .9895$).

```
. regress decom capacity years
```

Source	SS	df	MS	Number of obs = 5
Model	4666.16571	2	2333.08286	F(2, 2) = 189.42
Residual	24.6342883	2	12.3171442	Prob > F = 0.0053
Total	4690.8	4	1172.7	R-squared = 0.9947
				Adj R-squared = 0.9895
				Root MSE = 3.5096

decom	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
capacity	.1758739	.0247774	7.10	0.019	.0692653 .2824825
years	3.899314	.2643087	14.75	0.005	2.762085 5.036543
_cons	-11.39963	4.330311	-2.63	0.119	-30.03146 7.23219

The ado-file below defines program **ols0** using Mata commands. It simply calculates the vector of regression coefficients **b**. Mata commands start with **mata:** in this example. (Several other ways to use these commands interactively or in programs are described in the manuals.) The first two **mata:** commands define vector **y** and matrix **X** as “views” of the data in memory, specified by whatever left-hand-side (lhs) and right-hand-side (rhs) variables appeared in the **ols0** command line. A constant, 1, forms the last column of matrix **X**. **ols0** permits **in** or **if** qualifiers, or missing values. The estimating equation

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'\mathbf{y}$$

is written in Mata as

```
mata: b = invsym(X'X)*X'y
```

The fourth **mata:** command displays the resulting contents of **b**.

```
*! 21jun2012
!* L. Hamilton, Statistics with Stata (2012)
program ols0
    version 12.1
    syntax varlist(min=1 numeric) [in] [if]
    marksample touse
    gen cons_ = 1
    tokenize `varlist'
    local lhs "`1'"
    mac shift
    local rhs "`*'"
    mata: st_view(y=., ., "`lhs'", "`touse'") 
    mata: st_view(X=., ., (tokens("`rhs'"), "cons_"), "`touse'") 
    mata: b = invsym(X'X)*X'y
    mata: b
    drop cons_
end
```

Applied to the reactor decommissioning data, **ols0** obtains regression coefficients identical to those found earlier by **regress**.

```
. ols0 decom capacity years
      1
1 .1758738974
2 3.899313867
3 -11.39963279
```

Using Mata versions of the standard equations, program **ols1** (next page) adds the calculation of standard errors, *t* statistics, and *t* test probabilities. Again, the calculations lead to the same results we saw earlier with **regress**. Commas in the final **mata** statement of **ols1** are operators, meaning “join the columns of the following matrices.”

```
*! 21jun2012
!* L. Hamilton, Statistics with Stata (2012)
program ols1
    version 12.1
```

```

syntax varlist(min=1 numeric) [in] [if]
marksample touse
gen cons_ = 1
tokenize `varlist'
local lhs "`1"
mac shift
local rhs "`*"
mata: st_view(y=., ., "`lhs'", "`touse'")
mata: st_view(X=., ., (tokens("`rhs'"), "cons_"), "`touse'")
mata: b = invsym(X'X)*X'y
mata: e = y - X*b
mata: n = rows(X)
mata: k = cols(X)
mata: s2 = (e'e)/(n-k)
mata: V = s2*invsym(X'X)
mata: se = sqrt(diagonal(V))
mata: (b, se, b:/se, 2*ttail(n-k, abs(b:/se)))
drop cons_
end

. ols1 decom capacity years
      1          2          3          4
1   .1758738974   .0247774037   7.098156835   .0192756353
2   3.899313867    .26430873  14.75287581   .0045631637
3  -11.39963279   4.330310729  -2.632520735   .1190686843

```

We could expand this program to store results, and post them in a nicely-formatted output table similar to that of **regress**. Program **ols2** (next page) accomplishes something different, in order to demonstrate how Mata joins matrices together. It combines the numerical results seen above into a string matrix that also contains column headings and a list of independent-variable names. This happens through several additional **mata** commands. One defines row vector **vnames_** containing a list of variable names. The commas in this expression join three sets of columns: (1) the word “Yvar:” followed by the left-hand-side variable’s name; (2) the names of all right-hand-side variables; and (3) the word “_cons”.

```
mata: vnames_ = "Yvar: `lhs'", tokens("`rhs'"), "_cons"
```

The next long **mata** command uses within-line comment delimiters, /* and */, so that Mata reads past the end of two physical lines and sees this as all one command:

```
mata: vnames_!, ("Coef." \ strofreal(b)),      /*
*/ ("Std. Err." \ strofreal(se)),           /*
*/ ("t" \ strofreal(t)), ("P>|t|" \ strofreal(Prt))
```

The command displays a matrix in which the first column is the transpose of **vnames_** (that is, a column of variable names). The column of variable names is joined, using a comma, to a second column vector created with the word “Coefs” as its first row; remaining rows are filled by the coefficients in **b** converted from real numbers to strings. The backslash operator “\” joins rows to a matrix, just as “,” joins columns. The real-to-string conversion of **b** values is necessary to make the matrix types compatible. Similar operations in **ols2** form labeled columns of standard errors, *t* statistics, and probabilities.

```

*! 21jun2012
*! L. Hamilton, Statistics with Stata (2012)
program ols2
    version 12.1
    syntax varlist(min=1 numeric) [in] [if]
    marksample touse
    gen cons_ = 1
    tokenize `varlist'
    local lhs "`1'"
    mac shift
    local rhs "`*''"
    mata: st_view(y=., ., "`lhs'", "`touse'") 
    mata: st_view(X=., ., (tokens("`rhs'"), "cons_"), "`touse'") 
    mata: b = invsym(X'X)*X'y
    mata: e = y - X*b
    mata: n = rows(X)
    mata: k = cols(X)
    mata: s2 = (e'e)/(n-k)
    mata: V = s2*invsym(X'X)
    mata: se = sqrt(diagonal(V))
    mata: t = b:/se
    mata: Prt = 2*ttail(n-k, abs(b:/se))
    mata: vnames_ = "Yvar: `lhs'", tokens("`rhs'"), "_cons"
    mata: vnames_ , ("Coef." \ strofreal(b)), /*
        */ ("Std. Err." \ strofreal(se)), /*
        */ ("t" \ strofreal(t)), ("P>|t|" \ strofreal(Prt))
    drop cons_
end

. ols2 decom capacity years
      1          2          3          4          5
1  Yvar: decom      Coef.      Std. Err.          t      P>|t|
2    capacity     .1758739     .0247774     7.098157     .0192756
3    years       3.899314     .2643087    14.75288     .0045632
4    _cons      -11.39963     4.330311    -2.632521     .1190687

```

These Mata exercises, like other examples in this chapter, give only a glimpse of Stata programming. The *Stata Journal* publishes more inspired applications, and each update of Stata involves new or improved ado-files. Online NetCourses provide a guided route to fluency in writing your own programs.

Dataset Sources

The following publications or Web pages provide background information such as definitions, original sources and broader context of data used as examples in *Statistics with Stata* (2012). Often the example datasets are excerpts from larger files, or contain variables that have been merged from more than one source. See **References** for full bibliographic citations.

aids.raw

aids.dta

Selvin (1995)

Alaska_places.dta

Hamilton et al. (2011)

Alaska_regions.dta

Hamilton and Lammers (2011)

Antarctic2.dta

Milke and Heygster (2009)

Arctic9.dta

Sea ice extent: NSIDC (National Snow and Ice Data Center), Sea Ice Index.
http://nsidc.org/data/seaice_index/

Sea ice volume: PIOMAS (Pan-Arctic Ice Ocean Modeling and Assimilation System),
Polar Science Center, University of Washington. Arctic Sea Ice Volume Anomaly.
<http://psc.apl.washington.edu.wordpress/research/projects/arctic-sea-ice-volume-anomaly/>

Annual air temperature anomaly 64–90 °N: GISTEMP (GISS Surface Temperature Analysis), Goddard Institute for Space Studies, NASA.
<http://data.giss.nasa.gov/gistemp/>

attract2.dta

Hamilton (2003)

Canada1.dta

Canada2.dta

Federal, Provincial and Territorial Advisory Committee on Population Health (1996)

Climate.dta

NCDC global temperature: National Climatic Data Center, NOAA. Global Surface Temperature Anomalies. <http://www.ncdc.noaa.gov/cmb-faq/anomalies.php>
 NASA global temperature: GISTEMP (GISS Surface Temperature Analysis), Goddard Institute for Space Studies, NASA. <http://data.giss.nasa.gov/gistemp/>
 UAH global temperature: University of Alabama, Huntsville. <http://vortex.nsfc.uah.edu/data/msu/t2lt/uahncdc.lt>
 Aerosol Optical Depth (AOD): Sato et al. (1993). Goddard Institute for Space Studies, NASA. Forcings in GISS Climate Model. <http://data.giss.nasa.gov/modelforce/strataer/>
 Total Solar Irradiance (TSI): Fröhlich (2006). Physikalisch-Meteorologischen Observatoriums Davos, World Radiation Center (PMOD WRC). Solar Constant. <http://www.pmodwrc.ch/pmod.php?topic=tsi/composite/SolarConstant>
 Multivariate ENSO Index (MEI): Wolter and Timlin (1998). Earth Systems Research Laboratory, Physical Sciences Division, NOAA. Multivariate ENSO Index. <http://www.esrl.noaa.gov/psd/enso/mei/mei.html>
 Global average marine surface CO₂: Masarie and Tans (1995). Earth System Research Laboratory, Global Monitoring Division, NOAA. Trends in Atmospheric Carbon Dioxide. http://www.esrl.noaa.gov/gmd/ccgg/trends/global.html#global_data

election_2004i.dta

Robinson (2005). Geovisualization of the 2004 Presidential Election.
<http://www.personal.psu.edu/users/a/c/acr181/election.html>

electricity.dta

California Energy Commission (2012). U.S. Per Capita Electricity Use by State, 2010.
http://energyalmanac.ca.gov/electricity/us_per_capita_electricity-2010.html

*global1.dta**global2.dta**global3.dta**global_yearly.dta*

Multivariate ENSO Index (MEI): see *climate.dta*
 NCDC global temperature: see *climate.dta*

Granite2011_6.dta

Hamilton (2012). Also see “Do you believe the climate is changing?” by Hamilton (2011) at <http://www.carseyinstitute.unh.edu/publications/IB-Hamilton-Climate-Change-National-NH.pdf>

Greenland_sulfate.dta

Mayewski, Holdsworth et al. (1993); Mayewski, Meeker et al. (1993). Also see Sulfate and Nitrate Concentrations at GISP2 from 1750–1990.
<http://www.gisp2.sr.unh.edu/DATA/SO4NO3.html>

Greenland temperature.dta

GISP2 ice core temperature: Alley (2004). NOAA Paleoclimatology Program and World Data Center for Paleoclimatology, Boulder.
ftp://ftp.ncdc.noaa.gov/pub/data/paleo/icecore/greenland/summit/gisp2/isotopes/gisp2_temp_accum_alley2000.txt

Summit temperature 1987–1999: Shuman et al. (2001)

greenpop1.dta

Hamilton and Rasumssen (2010)

GSS_2010_SwS.dta

Davis et al. (2005). National Opinion Research Center (NORC), University of Chicago. General Social Survey. <http://www3.norc.org/GSS+Website/>

heart.dta

Selvin (1995)

*lakewin1.dta**lakewin2.dta**lakewin3.dta**lakesun.dta**lakesunwin.dta*

Lake Winnipesaukee ice out: <http://www.winnipesaukee.com/index.php?pageid=iceout>

Lake Sunapee ice out: http://www.town.sunapee.nh.us/Pages/SunapeeNH_Clerk/

Also see Hamilton et al. (2010a) at

http://www.carseyinstitute.unh.edu/publications/IB_Hamilton_Climate_Survey_NH.pdf

*MEI0.dta**MEI1.dta*

Multivariate ENSO Index: see *climate.dta*

MILwater.dta

Hamilton (1985)

*Nations2.dta**Nations3.dta*

Human Development Reports, United Nations Development Program. International Human Development Indicators. <http://hdrstats.undp.org/en/tables/>

oakridge.dta

Selvin (1995)

planets.dta

Beatty (1981)

PNWsurvey2_11.dta

Hamilton et al. (2010b, 2012). Also see “Ocean views” by Safford and Hamilton (2010),
at http://www.carseyinstitute.unh.edu/publications/PB_Safford_DowneastMaine.pdf

reactor.dta

Brown et al. (1986)

shuttle.dta

shuttle0.dta

Report of the Presidential Commission on the Space Shuttle Challenger Accident (1986)

Tufte (1997)

smoking1.dta

smoking1.dta

Rosner (1995)

snowfall.xls

Hamilton et al. (2003)

southmig1.dta

southmig2.dta

Voss et al. (2005)

student2.dta

Ward and Ault (1990)

whitemt1.dta

whitemt2.dta

Hamilton et al. (2007)

writing.dta

Nash and Schwartz (1987)

References

- Albright, J.J. and D.M. Marinova. 2010. "Estimating Multilevel Models Using SPSS, Stata, and SAS." Indiana University.
- Alley, R.B. 2004. GISP2 Ice Core Temperature and Accumulation Data. IGBP PAGES/World Data Center for Paleoclimatology Data Contribution Series #2004-013. NOAA/NGDC Paleoclimatology Program, Boulder CO, USA.
- Beatty, J.K., B. O'Leary and A. Chaikin (eds.). 1981. *The New Solar System*. Cambridge, MA: Sky.
- Belsley, D.A., E. Kuh and R.E. Welsch. 1980. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. New York: John Wiley & Sons.
- Box, G.E.P., G.M. Jenkins and G.C. Reinsel. 1994. *Time Series Analysis: Forecasting and Control*. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall.
- Brown, L.R., W.U. Chandler, C. Flavin, C. Pollock, S. Postel, L. Starke and E.C. Wolf. 1986. *State of the World 1986*. New York: W. W. Norton.
- California Energy Commission. 2012. "U.S. per capita electricity use by state in 2010." http://energyalmanac.ca.gov/electricity/us_per_capita_electricity-2010.html accessed 3/13/2012
- Chambers, J.M., W.S. Cleveland, B. Kleiner and P.A. Tukey. 1983. *Graphical Methods for Data Analysis*. Belmont, CA: Wadsworth.
- Chatfield, C. 2004. *The Analysis of Time Series: An Introduction*, 6th edition. Boca Raton, FL: CRC.
- Cleveland, W.S. 1993. *Visualizing Data*. Summit, NJ: Hobart Press.
- Cleves, M., W. Gould, R. Gutierrez and Y. Marchenko. 2010. *An Introduction to Survival Analysis Using Stata*, 3rd edition. College Station, TX: Stata Press.
- Cook, R.D. and S. Weisberg. 1982. *Residuals and Influence in Regression*. New York: Chapman & Hall.

- Cook, R.D. and S. Weisberg. 1994. *An Introduction to Regression Graphics*. New York: John Wiley & Sons.
- Cox, N.J. 2004a. "Stata tip 6: Inserting awkward characters in the plot." *Stata Journal* 4(1):95–96.
- Cox, N.J. 2004b. "Speaking Stata: Graphing categorical and compositional data." *Stata Journal* 4(2):190–215.
- Davis, J.A. T.W. Smith and P.V. Marsden. 2005. General Social Surveys, 1972–2004 Cumulative File [computer data file]. Chicago: National Opinion Research Center [producer]. Ann Arbor, MI: Inter-University Consortium for Political and Social Research [distributor].
- Diggle, P.J. 1990. *Time Series: A Biostatistical Introduction*. Oxford: Oxford University Press.
- Enders, W. 2004. *Applied Econometric Time Series*, 2nd edition. New York: John Wiley & Sons.
- Everitt, B.S., S. Landau and M. Leese. 2001. *Cluster Analysis*, 4th edition. London: Arnold.
- Federal, Provincial and Territorial Advisory Commission on Population Health. 1996. *Report on the Health of Canadians*. Ottawa: Health Canada Communications.
- Foster, G. and S. Rahmstorf. 2011. "Global temperature evolution 1979–2010." *Environmental Research Letters* 6. DOI:10.1088/1748-9326/6/4/044022
- Fox, J. 1991. *Regression Diagnostics*. Newbury Park, CA: Sage Publications.
- Fröhlich, C. 2006. "Solar irradiance variability since 1978—Revision of the PMOD composite during solar cycle 21." *Space Science Review* 125:53–65.
- Gould, W., J. Pitblado and B. Poi. 2010. *Maximum Likelihood Estimation with Stata*, 4th edition. College Station, TX: Stata Press.
- Hamilton, D.C. 2003. "The Effects of Alcohol on Perceived Attractiveness." Senior Thesis. Claremont, CA: Claremont McKenna College.
- Hamilton, J.D. 1994. *Time Series Analysis*. Princeton, NJ: Princeton University Press.
- Hamilton, L.C. 1985. "Who cares about water pollution? Opinions in a small-town crisis." *Sociological Inquiry* 55(2):170–181.
- Hamilton, L.C. 1992a. *Regression with Graphics: A Second Course in Applied Statistics*. Pacific Grove, CA: Brooks/Cole.
- Hamilton, L.C. 1992b. "Quartiles, outliers and normality: Some Monte Carlo results." Pp. 92–95 in J. Hilbe (ed.) *Stata Technical Bulletin Reprints, Volume 1*. College Station, TX: Stata Press.

- Hamilton, L.C., D.E. Rohall, B.C. Brown, G. Hayward and B.D. Keim. 2003. "Warming winters and New Hampshire's lost ski areas: An integrated case study." *International Journal of Sociology and Social Policy* 23(10):52–73.
- Hamilton, L.C., B.C. Brown and B.D. Keim. 2007. "Ski areas, weather and climate: Time series models for New England case studies." *International Journal of Climatology* 27:2113–2124.
- Hamilton, L.C. and R.O. Rasmussen. 2010. "Population, sex ratios and development in Greenland." *Arctic* 63(1):43–52.
- Hamilton, L.C., B.D. Keim and C.P. Wake. 2010a. "Is New Hampshire's climate warming?" New England Policy Brief No. 4. Durham, NH: Carsey Institute, University of New Hampshire.
- Hamilton, L.C., C.R. Colocousis and C.M. Duncan. 2010b. "Place effects on environmental views." *Rural Sociology* 75(2):326–347.
- Hamilton, L.C. and R.B. Lammers. 2011. "Linking pan-Arctic human and physical data." *Polar Geography* 34(1–2):107–123.
- Hamilton, L.C., D.M. White, R.B. Lammers and G. Myerchin. 2011. "Population, climate and electricity use in the Arctic: Integrated analysis of Alaska community data." *Population and Environment* 33(4):269–283. DOI: 10.1007/s11111-011-0145-1.
- Hamilton, L.C. 2012. "Did the Arctic ice recover? Demographics of true and false climate facts." Paper presented at the American Sociological Association. Denver, Colorado, August 17–20.
- Hamilton, L.C., T.G. Safford and J.D. Ulrich. 2012. "In the wake of the spill: Environmental views along the Gulf Coast." *Social Science Quarterly* DOI: 10.1111/j.1540-6237.2012.00840.x
- Hardin, J. and J. Hilbe. 2012. *Generalized Linear Models and Extensions*, 3rd edition. College Station, TX: Stata Press.
- Hoaglin, D.C., F. Mosteller and J.W. Tukey (eds.). 1983. *Understanding Robust and Exploratory Data Analysis*. New York: John Wiley & Sons.
- Hoaglin, D.C., F. Mosteller and J.W. Tukey (eds.). 1985. *Exploring Data Tables, Trends and Shapes*. New York: John Wiley & Sons.
- Hosmer, D.W., Jr., S. Lemeshow and S. May. 2008. *Applied Survival Analysis: Regression Modeling of Time to Event Data*, 2nd edition. New York: John Wiley & Sons.
- Hosmer, D.W., Jr. and S. Lemeshow. 2000. *Applied Logistic Regression*, 2nd edition. New York: John Wiley & Sons.
- Kline, R.B. 2010. *Principles and Practice of Structural Equation Modeling*, Third Edition. New York: Guilford.

- Korn, E.L. and B.I. Graubard. 1999. *Analysis of Health Surveys*. New York: Wiley.
- Lean, J.L. and D.H. Rind. 2008. "How natural and anthropogenic influences alter global and regional surface temperatures: 1889 to 2006." *Geophysical Research Letters* 35 DOI:10.1029/2008GL034864
- Lee, E.T. 1992. *Statistical Methods for Survival Data Analysis*, 2nd edition. New York: John Wiley & Sons.
- Lee, E.S. and R.N. Forthofer. 2006. *Analyzing Complex Survey Data*, second edition. Thousand Oaks, CA: Sage.
- Levy, P.S. and S. Lemeshow. 1999. *Sampling of Populations: Methods and Applications*, 3rd Edition. New York: Wiley.
- Li, G. 1985. "Robust regression." Pp. 281–343 in D. C. Hoaglin, F. Mosteller and J. W. Tukey (eds.) *Exploring Data Tables, Trends and Shapes*. New York: John Wiley & Sons.
- Long, J.S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage Publications.
- Long, J. S. and J. Freese. 2006. *Regression Models for Categorical Dependent Variables Using Stata*, 2nd edition. College Station, TX: Stata Press.
- Luke, D.A. 2004. *Multilevel Modeling*. Thousand Oaks, CA: Sage.
- Mallows, C.L. 1986. "Augmented partial residuals." *Technometrics* 28:313–319.
- Masarie, K.A. and P.P. Tans. 1995. "Extension and integration of atmospheric carbon dioxide data into a globally consistent measurement record." *Journal of Geophysical Research* 100:11593–11610.
- Mayewski, P.A., G. Holdsworth, M.J. Spencer, S. Whitlow, M. Twickler, M.C. Morrison, K.K. Ferland and L.D. Meeker. 1993. "Ice-core sulfate from three northern hemisphere sites: Source and temperature forcing implications." *Atmospheric Environment* 27A(17/18):2915–2919.
- Mayewski, P.A., L.D. Meeker, S. Whitlow, M.S. Twickler, M.C. Morrison, P. Bloomfield, G.C. Bond, R.B. Alley, A.J. Gow, P.M. Grootes, D.A. Meese, M. Ram, K.C. Taylor and W. Wumkes. 1994. "Changes in atmospheric circulation and ocean ice cover over the North Atlantic during the last 41,000 years." *Science* 263:1747–1751.
- McCullagh, P. and J.A. Nelder. 1989. *Generalized Linear Models*, 2nd edition. London: Chapman & Hall.
- McCulloch, C.E. and S.R. Searle. 2001. *Generalized, Linear, and Mixed Models*. New York: Wiley.

- Milke, A., and G. Heygster. 2009. "Trend der Meereisausdehnung von 1972–2009." Technical Report, Institute of Environmental Physics, University of Bremen, August 2009, 41 pages. http://www.iup.uni-bremen.de/iuppage/psa/documents/Technischer_Bericht_Milke_2009.pdf
- Mitchell, M.N. 2008. *A Visual Guide to Stata Graphics*, 2nd edition. College Station, TX: Stata Press.
- Mitchell, M.N. 2012. *Interpreting and Visualizing Regression Models Using Stata*. College Station, TX: Stata Press.
- Moore, D. 2008. *The Opinion Makers: An Insider Reveals the Truth about Opinion Polls*. Boston: Beacon Press.
- Nash, J. and L. Schwartz. 1987. "Computers and the writing process." *Collegiate Microcomputer* 5(1):45–48.
- Rabe-Hesketh, S. and B. Everitt. 2000. *A Handbook of Statistical Analysis Using Stata*, 2nd edition. Boca Raton, FL: Chapman & Hall.
- Rabe-Hesketh, S. and A. Skrondal. 2012. *Multilevel and Longitudinal Modeling Using Stata*, 3rd edition. College Station, TX: Stata Press.
- Raudenbush, S.W. and A.S. Bryk. 2002. *Hierarchical Linear Models: Applications and Data Analysis Methods*, 2nd edition. Newbury Park, CA: Sage.
- Raudenbush, S.W., A.S. Bryk, Y.F. Cheong & R. Congdon. 2005. *HLM 5: Hierarchical Linear and Nonlinear Modeling*. Lincolnwood, IL: Scientific Software International.
- Report of the Presidential Commission on the Space Shuttle Challenger Accident. 1986. Washington, DC.
- Robinson, A. 2005. "Geovisualization of the 2004 presidential election." Available at <http://www.personal.psu.edu/users/a/c/acr181/election.html> (accessed 3/8/2008).
- Rosner, B. 1995. *Fundamentals of Biostatistics*, 4th edition. Belmont, CA: Duxbury Press.
- Safford, T.G. and L.C. Hamilton. 2010. "Ocean views: Coastal environmental problems as seen by Downeast Maine residents." *New England Policy Brief No. 3*. Durham, NH: Carsey Institute, University of New Hampshire.
- Sato, M., J.E. Hansen, M.P. McCormick and J.B. Pollak. 1993. "Stratospheric aerosol optical depths, 1850–1990." *Journal of Geophysical Research* 98:22,987–22,994.
- Selvin, S. 1995. *Practical Biostatistical Methods*. Belmont, CA: Duxbury Press.
- Selvin, S. 1996. *Statistical Analysis of Epidemiologic Data*, 2nd edition. New York: Oxford University.

- Shuman, C.A., K. Steffen, J.E. Box and C.R. Stearns. 2001. "A Dozen years of temperature observations at the summit: Central Greenland automatic weather stations 1987–99." *Journal of Applied Meteorology*, 40:741–752.
- Shumway, R.H. 1988. *Applied Statistical Time Series Analysis*. Upper Saddle River, NJ: Prentice-Hall.
- Skrondal, A. and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman & Hall/CRC.
- StataCorp. 2011. *Getting Started with Stata for Macintosh*. College Station, TX: Stata Press.
- StataCorp. 2011. *Getting Started with Stata for Unix*. College Station, TX: Stata Press.
- StataCorp. 2011. *Getting Started with Stata for Windows*. College Station, TX: Stata Press.
- StataCorp. 2011. *Mata Reference Manual* (2 volumes). College Station, TX: Stata Press.
- StataCorp. 2011. *Stata Base Reference Manual* (3 volumes). College Station, TX: Stata Press.
- StataCorp. 2011. *Stata Data Management Reference Manual*. College Station, TX: Stata Press.
- StataCorp. 2011. *Stata Graphics Reference Manual*. College Station, TX: Stata Press.
- StataCorp. 2011. *Stata Programming Reference Manual*. College Station, TX: Stata Press.
- StataCorp. 2011. *Stata Longitudinal/Panel Data Reference Manual*. College Station, TX: Stata Press.
- StataCorp. 2011. *Stata Multivariate Statistics Reference Manual*. College Station, TX: Stata Press.
- StataCorp. 2011. *Stata Quick Reference and Index*. College Station, TX: Stata Press.
- StataCorp. 2011. *Stata Structural Equation Reference Manual*. College Station, TX: Stata Press.
- StataCorp. 2011. *Stata Survey Data Reference Manual*. College Station, TX: Stata Press.
- StataCorp. 2011. *Stata Survival Analysis and Epidemiological Tables Reference Manual*. College Station, TX: Stata Press.
- StataCorp. 2011. *Stata Time-Series Reference Manual*. College Station, TX: Stata Press.
- StataCorp. 2011. *Stata User's Guide*. College Station, TX: Stata Press.
- Street, J.O., R.J. Carroll and D. Ruppert. 1988. "A note on computing robust regression estimates via iteratively reweighted least squares." *The American Statistician* 42(2):152–154.

- Tufte, E.R. 1990. *Envisioning Information*. Cheshire CT: Graphics Press.
- Tufte, E.R. 1997. *Visual Explanations: Images and Quantities, Evidence and Narrative*. Cheshire CT: Graphics Press.
- Tufte, E.R. 2001. *The Visual Display of Quantitative Information*, 2nd edition. Cheshire CT: Graphics Press.
- Tufte, E.R. 2006. *Beautiful Evidence*. Cheshire CT: Graphics Press.
- Tukey, J.W. 1977. *Exploratory Data Analysis*. Reading, MA: Addison-Wesley.
- Velleman, P.F. 1982. “Applied Nonlinear Smoothing,” pp.141–177 in Samuel Leinhardt (ed.) *Sociological Methodology 1982*. San Francisco: Jossey-Bass.
- Velleman, P.F. and D.C. Hoaglin. 1981. *Applications, Basics, and Computing of Exploratory Data Analysis*. Boston: Wadsworth.
- Verbeke, G. and G. Molenberghs. 2000. *Linear Mixed Models for Longitudinal Data*. New York: Springer.
- Voss, P.R., S. McNiven, R.B. Hammer, K.M. Johnson and G.V. Fugitt. 2005. “County-specific net migration by five-year age groups, Hispanic origin, race, and sex, 1990–2000.” Ann Arbor, MI: Inter-university Consortium for Political and Social Research, 2005-05-23.
- Ward, S. and S. Ault. 1990. “AIDS knowledge, fear, and safe sex practices on campus.” *Sociology and Social Research* 74(3):158–161.
- White, J.W.C., R.B. Alley, J. Brigham-Grette, J.J. Fitzpatrick, A.E. Jennings, S.J. Johnsen, G.H. Miller, R.S. Nerem and L. Polyak. 2010. “Past rates of climate change in the Arctic.” *Quaternary Science Reviews* 29(15–16):1716–1727.
- Wild, M., A. Ohmura and K. Makowski. 2007. “Impact of global dimming and brightening on global warming,” *Geophysical Research Letters*. DOI:10.1029/2006GL028031.
- Wolter, K. and M.S. Timlin. 1998. “Measuring the strength of ENSO events—how does 1997/98 rank?” *Weather* 53:315–324.

Index

A

acprplot (augmented component-plus-residual plot), 165, 212
added-variable plot, 166, 210
ado-file (automatic do), 424
alpha (Cronbach's alpha reliability), 313
analysis of covariance (ANCOVA), 155
analysis of variance (ANOVA), 143, 151
append, 46
ARCH model (autoregressive conditional heteroskedasticity), 351
area plot, 101
args (arguments in program), 431
ARIMA model (autoregressive integrated moving average), 351, 374
arithmetic operator, 29
ARMAX model (time series regression), 352, 374, 382
ASCII
 characters, 82
 data, 44
autocode (create ordinal variables), 41
autocorrelation, 208

B

band regression, 216
bar chart, 87, 116, 434
Bartlett's test for equal variances, 151
beta weight (standardized regression coefficient), 179
Bonferroni multiple-comparison test
 correlation matrix, 171
 one-way ANOVA, 153
bootstrap, 309, 447
Box–Cox transformation, 132, 238
box plot, 71

C

catplot, 116, 272, 435
centering to reduce multicollinearity, 189
chi-squared
 deviance (logit regression), 260
 in cross-tabulation, 133
 likelihood-ratio, 108, 134, 261
 probability plot, 93
classification table (logit regression), 259
cluster analysis, 327
communality (factor analysis), 323
component-plus-residual plot, 212
conditional effect plot, 262
confidence interval
 binomial, 127
 bootstrap, 310
 mean, 125, 145, 159
 median (.5 quantile), 227
 percentage, 108, 116
 Poisson, 127
 regression, 77, 169, 188, 205
 robust mean, 227
 robust variance, 190
Cook's *D*, 197, 201
correlation
 Kendall's tau, 134, 174
 Pearson product-moment, 170
 random data, 60
 regression estimates, 172
 Spearman, 173
covariance, 173
COVRATIO, 164, 202,
Cox proportional hazard, 293
Cramér's *V*, 124
Cronbach's alpha, 313
cross-correlation, 373
cross-tabulation, 133

D

database file, 45
 dates, 33, 354
 dendrogram, 315, 328
 DFBETA, 199
 Dickey–Fuller test, 376
 difference (time series), 367
 do-file, 61, 95, 423
 Doornik–Hansen omnibus test, 130
 dot chart, 67, 152
drawnorm (normal variable), 60
 dummy variable, 39, 166, 181
 Durbin–Watson test, 208, 371

E

egen, 35
 eigenvalue, 316
 empirical orthogonal function (EOF), 323
 epidemiological tables, 123
 error-bar chart, 158
 event-count model, 283
 Exploratory Data Analysis (EDA), 125
 exponential filter (time series), 356
 exponential growth model, 231
 exponential regression (survival analysis), 299

F

factor analysis, 313
 filter (time series), 353
 Fisher's exact test in cross-tabulation, 124
 fixed and random effects, 387
foreach, 430
 format
 label in graph, 88, 117, 300, 357
 date, 33, 255, 353
 input data, 44
 numeric display, 26
forvalues, 429
 frequency table, 133
 frequency weights (**fweight**), 55, 66, 75, 140
 functions, 32

G

generalized linear models (GLM), 307
generate, 26

gladder, 131

glamm, 388
 Gompertz growth model, 232
 Goodman and Kruskal's gamma, 134
 Graph Editor, 97
gsort (general sorting), 16, 431

H

hat matrix, 199
 hazard function, 285, 296, 300
 help file, 7, 441
 heteroskedasticity, 165, 190, 202
histogram, 68
 HLM (hierarchical linear modeling), 387
 Holt–Winters smoothing, 356
 Huber/White robust standard errors, 190

I

if qualifier, 23
if...else, 430
 import data, 42
 importance weights (**iweight**), 55
in qualifier, 292
 incidence rate, 288, 292, 303
infile (read ASCII data), 15, 287
infix (read fixed-format data), 44
 influence
 logistic regression, 260, 264
 regression (OLS), 197, 211
insheet (read spreadsheet data), 22, 44, 61
 interaction effect
 ANOVA, 144, 155
 regression, 185
 interquartile range (IQR), 51, 72, 129
 iteratively reweighted least squares (IRLS), 224

J

jackknife, 308
 jitter, 183

K

Kaplan–Meier survivor function, 290
 Kendall's tau, 134
 kernel density, 66
 Kruskal–Wallis test, 143, 153
 kurtosis, 126, 129

L

L-estimator, 224
label data, 20
label define, 28
label values, 28
label variable, 20
ladder of powers, 130
lag (time series), 366
lead (time series), 366
legend in graph, 54, 76
letter-value display, 128
leverage, 165, 199
leverage-vs.-squared-residuals plot, 213
likelihood-ratio chi-squared, 108, 134, 261
line plot, 80
link function (GLM), 307
Ljung–Box portmanteau Q, 353, 372
log file, 3
logical operator, 24
logistic growth model, 216,
logistic regression, 251, 415
looping, 429
lowess smoothing, 94, 165, 195, 217
Iroc (logistic ROC), 253
Irtest (likelihood-ratio test), 261, 269, 388
Isens (logistic sensitivity graph), 253

M

M-estimator, 224
macro, 426
Macro
Mann–Whitney *U* test, 150
margins, 159, 178, 262, 275, 408
marginsplot, 159, 178, 262, 275, 408
marker label in graph, 72, 77, 144, 210
marker symbol in graph, 75, 183, 446
marksample, 433
mata (matrix programming), 452
matched-pairs test, 145
mean
 arithmetic, 2, 125
 confidence interval, 127
 robust, 227
 weighted, 57, 108
measurement model, 344
median, 49, 67, 72, 88, 91, 126, 227, 445
median regression, 221, 227
merge, 47

missing value

 codes, 29
 multiple imputation, 241, 278
mixed effects, 387
Monte Carlo, 445
moving average, 351
multicollinearity, 202
multilevel modeling (mixed effects), 387
multinomial logistic regression, 270
multiple-comparison test
 correlation matrix, 171
one-way ANOVA, 153

N

negative binomial regression, 286, 308, 387
negative exponential growth model, 232
nonlinear regression, 230
nonlinear smoothing, 353
normal distribution
 artificial data, 60
 curve, 65, 90
 test for, 92, 129
normal probability plot, 92

O

ODBC (Open Database Connectivity), 45
odds ratio, 258
omitted-variables test, 165
one-sample *t* test, 145
one-way ANOVA, 151
order
 observations (**sort**), 20
 variables (**order**), 21
ordered logistic regression, 268
outfile (write ASCII data), 45
outlier, 71, 78, 126, 166, 200, 222, 268,
 322, 446
overlay twoway graphs, 74

P

paired-difference tests, 145
panel data, 252, 353, 387, 413
partial autocorrelation, 372
partial regression plot, 166, 210
Pearson correlation, 170
percentiles, 50, 67, 88, 126
periodogram, 352

- Phillips–Perron test, 376
- pie chart, 87
- poisgof** (Poisson goodness of fit test), 305
- Poisson regression, 303, 307
- polynomial regression, 85
- population pyramid, 103
- poststratification, 113
- predict** (predicted values, residuals, etc.)
 - anova**, 158
 - arima**, 378, 383
 - Cox proportional hazard, 296
 - factor scores, 321, 337
 - logistic**, 252, 258, 264
 - multinomial logit, 253
 - nonlinear, 231
 - ordered logit, 269
 - regress**, 192, 197
 - rreg**, 223
 - xtmelogit**, 418
 - xtmixed**, 394, 402, 411
- principal component analysis, 313
- print graph, 6, 69
- print results, 5, 7
- probability weights (**pweight**), 55, 108
- probit regression, 251
- promax rotation, 318
- pwcorr** (pairwise correlation), 171

- Q**
- qladder**, 132
- quantile
 - plots, 90, 132
 - regression, 221, 227
- quartile, 72, 92, 126, 217
- quietly**, 115, 201

- R**
- r-class, 44
- Ramsey specification error test (RESET), 165
- random data, 57, 445
- random effects, 387
- random numbers and sampling, 57

- range
 - axis in graph, 102
 - cells in spreadsheet, 15, 42
 - plots, 85, 99, 159
 - standardization, 332
- rank
 - rank tests, 145, 148
 - transformation, 36
- real** function, 38
- regression, 163, 215, 251, 283, 313, 387
- relational operator, 24
- relative risk ratio, 274
- rename**, 17
- replace**, 26
- reshape**, 52
- residual, 192
- residual-vs.-fitted plot, 195
- robust
 - regression, 221, 227
 - standard errors and variance, 190
- rotation (factor analysis), 318
- rough, 353

- S**
- sample** (draw random sample), 6
- sampling weights, 107
- sandwich estimator of variance, 190
- SAS data files, 43, 45,
- saveold** (save dataset in previous Stata format), 16
- scatterplot, 74
- Scheffé multiple-comparison test, 153, 171
- screeplot** (scree plot of eigenvalues), 317
- search**, 8
- seasonal (time series), 352, 356, 367, 376
- serrbar** (standard-error bar plot), 145
- Shapiro–Francia, Shapiro–Wilk tests, 130
- Šidák multiple-comparison test, 153, 171
- sign test, 145
- signed-rank test, 148
- skewness, 126, 129, 193
- SMCL (Stata Markup and Control Language), 3, 73, 442
- smoothing
 - lowess, 217
 - nonlinear, 353
 - time series, 353

- sort, 20
- Spearman correlation, 173
- spectral density, 352
- spike plot, 50, 360
- spreadsheet data, 15, 21, 42
- SPSS data files, 45
- standard deviation, 50, 59, 123
- standardized regression coefficient, 179, 245
- standardized variable, 35
- Stat/Transfer, 45
- Stata Journal*, 10
- stationary time series, 376
- statsby**, 51
- stcox** (Cox hazard model), 294
- stcurve** (survival analysis graphs), 301
- stem-and-leaf display, 127
- stepwise regression, 166
- stphplot**, 285
- streg** (survival analysis regression), 299
- string variable, 18, 36
- structural equation modeling, 244, 344
- sts generate** (survivor function), 285, 300
- sts graph** (survivor function), 285, 291
- sts test** (test survivor function), 285, 293
- studentized residual, 197
- subscript
 - graphs, 73
 - variables, 41
- summarize** (summary statistics), 2, 125
- survey data, 107
- survival analysis, 283
- symmetry plot, 90
- syntax** (programming), 432

- T**
- t test**
 - means, 145
 - regression, 169
- table**, 124, 137
- tabstat**, 126
- tabulate**, 25, 30, 39, 108, 115, 123
- technical support, 9
- test** (hypothesis test for model), 180, 306
- text in graph, 93
- time plot, 80, 351
- time series, 351

- tobit regression, 352, 387
- tree diagram, 328, 330, 334
- tsline** (time plots), 355
- tssmooth** (time series smoothing), 353
- two-sample test, 148

- U**
- unequal variance in *t* test, 150
- unit root, 376

- V**
- variance inflation factor (VIF), 202
- varimax rotation, 318

- W**
- Weibull regression (survival analysis), 299
- weighted least squares (WLS), 55, 166, 224
- weights, 55, 76, 107, 140, 199, 224, 271
- Welsch's W, 164, 201
- which**, 424
- while**, 430
- white noise, 372, 379
- Wilcoxon rank-sum test, 145, 150
- Wilcoxon signed-rank test, 145, 148
- wntestq** (Ljung–Box white noise *Q*), 379

- X**
- xcorr** (cross-correlation), 353, 373
- xtmelogit**, 415
- xtmepoisson**, 389
- xtmixed** (multilevel mixed-effect models), 387

- Z**
- z* score (standardized variable), 35

