8

Fitting Curves

Basic regression and correlation methods assume linear relationships. Linear models provide reasonable and simple approximations for many real phenomena, over a limited range of values. But analysts also encounter phenomena where linear approximations are too simple; these call for nonlinear alternatives. This chapter describes three broad approaches to modeling nonlinear or curvilinear relationships:

1. Nonparametric methods, including band regression and lowess smoothing.
2. Linear regression with transformed variables ("curvilinear regression"), including Box–Cox methods.
3. Nonlinear regression.

Nonparametric regression serves as an exploratory tool because it can summarize data patterns visually without requiring the analyst to specify a particular model in advance. Transformed variables extend the usefulness of linear parametric methods, such as OLS regression (**`regress`**), to encompass curvilinear relationships as well. Nonlinear regression, on the other hand, requires a different class of methods that can estimate parameters of intrinsically nonlinear models.

The following menu groups cover many of the operations discussed in this chapter. The final topic, nonlinear regression, requires a command-based approach.

Graphics – Twoway

Statistics – Nonparametric analysis – Lowess smoothing

Data – Create or change variables – Create new variable

Statistics – Linear regression and related

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 by the same or different parameters, 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. Lowess calculations use a bandwidth of .4 (40% of the data). In order to calculate and keep the smoothed values as a new variable, use the related command **lowess**.

```
. 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 more options than **graph twoway lowess**, including fitting methods and the ability to save predicted values. See **help lowess** for details.

```
. nl exp2 y x
```

Uses iterative nonlinear least squares to fit a 2-parameter exponential growth model,

predicted $y = b_1 b_2{}^x$

The term **exp2** refers to a separate program that specifies the model itself. You can write a program to define your own model, or use one of the common models (including exponential, logistic, and Gompertz) supplied with Stata. After **nl**, use **predict** to generate predicted values or residuals.

```
. nl log4 y x, init(B0=5, B1=25, B2=.1, B3=50)
```

Fits a 4-parameter logistic growth model (**log4**) of the form

predicted $y = b_0 + b_1/(1 + \exp(-b_2(x - b_3)))$

Sets initial parameter values for the iterative estimation process at $b_0 = 5$, $b_1 = 25$, $b_2 = .1$, and $b_3 = 50$.

```
. regress lny x1 sqrtx2 invx3
```

Performs curvilinear regression using the variables *lny*, *x1*, *sqrtx2*, and *invx3*. These variables were previously generated by nonlinear transformations of the raw variables *y*, *x2*, and *x3* through commands such as the following:

```
. generate lny = ln(y)
. generate sqrtx2 = sqrt(x2)
. generate invx3 = 1/x3
```

When, as in this example, the y variable was transformed, the predicted values generated by **predict yhat**, or residuals generated by **predict e, resid**, will be also in transformed units. For graphing or other purposes, we might want to return predicted values or residuals to raw-data units, using inverse transformations such as

```
. replace yhat = exp(yhat)
```

Band Regression

Nonparametric regression methods generally do not yield an explicit regression equation. They are primarily graphic tools for displaying the relationship, possibly nonlinear, between *y* and *x*. Stata can draw a simple kind of nonparametric regression, band regression, onto any scatterplot or scatterplot matrix. For illustration, consider these sobering Cold War data (*missile.dta*) from MacKenzie (1990). The observations are 48 types of long-range nuclear missiles, deployed by the U.S. and Soviet Union during their arms race, 1958 to 1990:

```
Contains data from C:\data\missile.dta
  obs:            48                          Missiles (MacKenzie 1990)
 vars:             6                          16 Jul 2005 14:57
 size:         1,392 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
missile         str15  %15s                   Missile
country         byte   %8.0g       soviet     US or Soviet missile?
year            int    %8.0g                  Year of first deployment
type            byte   %8.0g       type       ICBM or submarine-launched?
range           int    %8.0g                  Range in nautical miles
CEP             float  %9.0g                  Circular Error Probable (miles)
-------------------------------------------------------------------------------
Sorted by:  country  year
```

Variables in *missile.dta* include an accuracy measure called the "Circular Error Probable" (*CEP*). *CEP* represents the radius of a bulls eye within which 50% of the missile's warheads should land. Year by year, scientists on both sides worked to improve accuracy (Figure 8.1).

```
. graph twoway mband CEP year, bands(8)
      ||  scatter CEP year
      ||  , ytitle("Circular Error Probable, miles") legend(off)
```

Figure 8.1

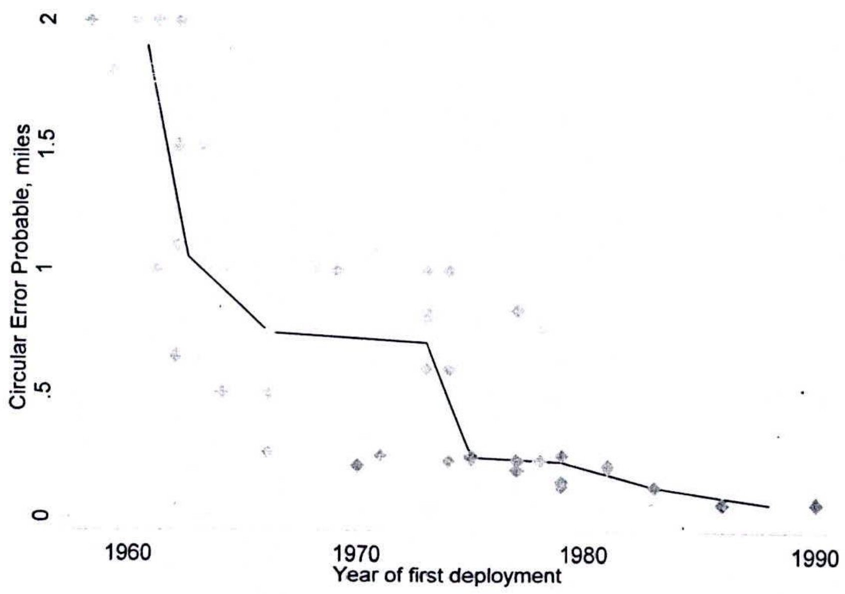
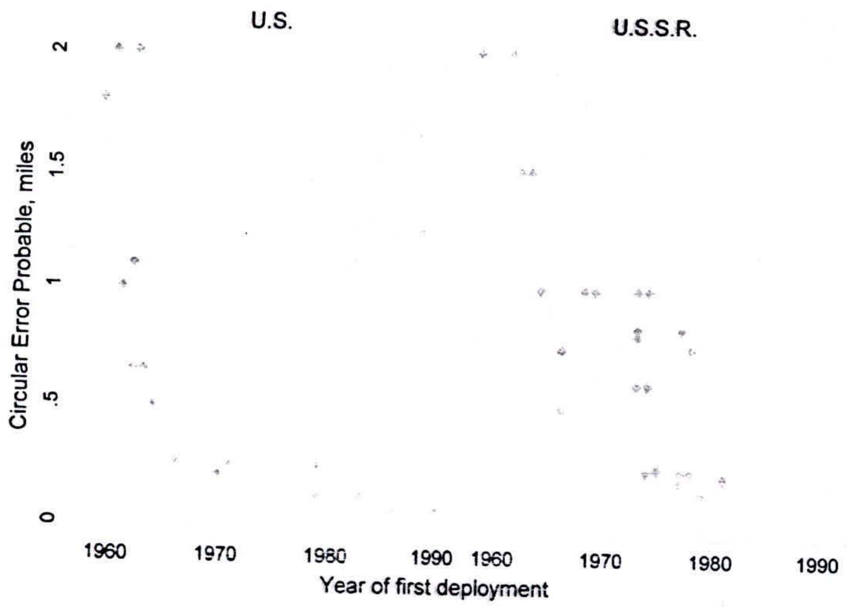


Figure 8.1 shows *CEP* declining (accuracy increasing) over time. The option **bands(8)** instructs **graph twoway mband** to divide the scatterplot into 8 equal-width vertical bands and draw line segments connecting the points (median *x*, median *y*) within each band. This curve traces how the median of *CEP* changes with *year*.

Nonparametric regression does not require the analyst to specify a relationship's functional form in advance. Instead, it allows us to explore the data with an "open mind." This process often uncovers interesting results, such as when we view trends in U.S. and Soviet missile accuracy separately (Figure 8.2). The **by(*country*)** option in the following command produces separate plots for each country, each with overlaid band-regression curve and scatterplot. Within the `by( )` option are suboptions controlling the legend and note.

```
. graph twoway mband CEP year, bands(8)
      ||  scatter CEP year
      ||  , ytitle("Circular Error Probable, miles")
        by(country, legend(off) note(""))
```

Figure 8.2



The shapes of the two curves in Figure 8.2 differ substantially. U.S. missiles became much more accurate in the 1960s, permitting a shift to smaller warheads. Three or more small warheads would fit on the same size missile that formerly carried one large warhead. The accuracy of Soviet missiles improved more slowly, apparently stalling during the late 1960s to early 1970s, and remained a decade or so behind their American counterparts. To make up for this accuracy disadvantage, Soviet strategy emphasized larger rockets carrying high-yield warheads. Nonparametric regression can assist with a qualitative description of this sort or serve as a preliminary to fitting parametric models such as those described later.

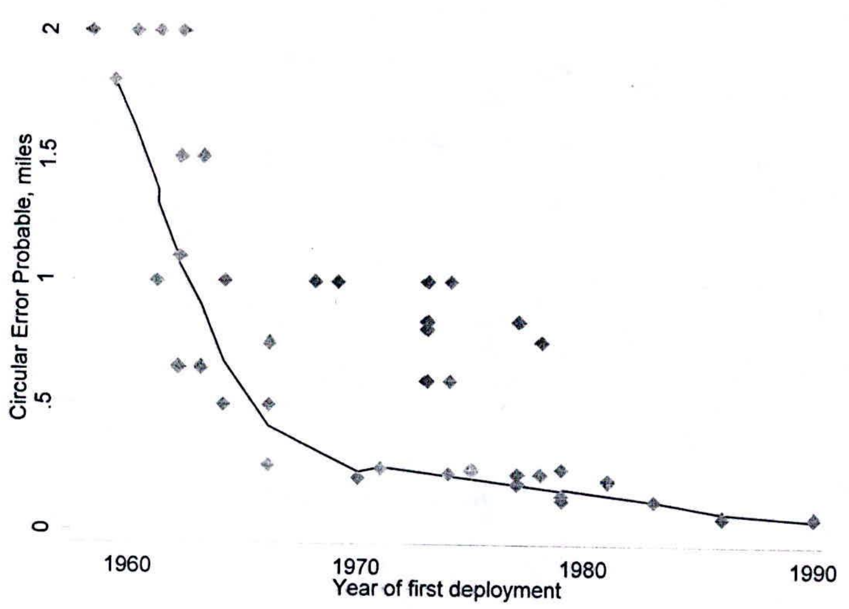
We can add band regression curves to any scatterplot by overlaying an **mband** (or **mspline**) plot. Band regression's simplicity makes it a convenient exploratory tool, but it possesses one notable disadvantage — the bands have the same width across the range of *x* values, although some of these bands contain few or no observations. With normally distributed variables, for example, data density decreases toward the extremes. Consequently, the left and right endpoints of the band regression curve (which tend to dominate its appearance) often reflect just a few data points. The next section describes a more sophisticated, computation-intensive approach.

Lowess Smoothing

The **lowess** and **graph twoway lowess** commands accomplish a form of nonparametric regression called lowess smoothing (for **lo**cally **we**ighted **s**catterplot **s**moothing). In general the **lowess** command is more specialized and more powerful, with options that control details of the fitting process. **graph twoway lowess** has advantages of simplicity, and follows the familiar syntax of the **graph twoway** family. The following example uses **graph twoway lowess** to plot *CEP* against *year* for U.S. missiles only (*country* == 0).

```
. graph twoway lowess CEP year if country == 0, bwidth(.4)
        ||  scatter CEP year
        ||  , legend(off) ytitle("Circular Error Probable, miles")
```

Figure 8.3



A graph very similar to Figure 8.2 would result if we had typed instead

```
. lowess CEP year if country == 0, bwidth(.4)
```

Like Figure 8.2, Figure 8.3 (next page) shows U.S. missile accuracy improving rapidly during the 1960s and progressing at a more gradual rate in the 1970s and 1980s. Lowess-smoothed values of *CEP* are generated here with the name *lsCEP*. The **bwidth(.4)** option specifies the lowess bandwidth: the fraction of the sample used in smoothing each point. The default is **bwidth(.8)**. The closer bandwidth is to 1, the greater the degree of smoothing.

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^* 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 jth 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.
`plot( )`	Provides a way to add other plots to the generated graph; see **`help`** `plot_option`.
`rlopts()`	Affects the rendition of the reference line; see **`help cline_options`**.

Because it requires n weighted regressions, lowess smoothing proceeds slowly with large samples.

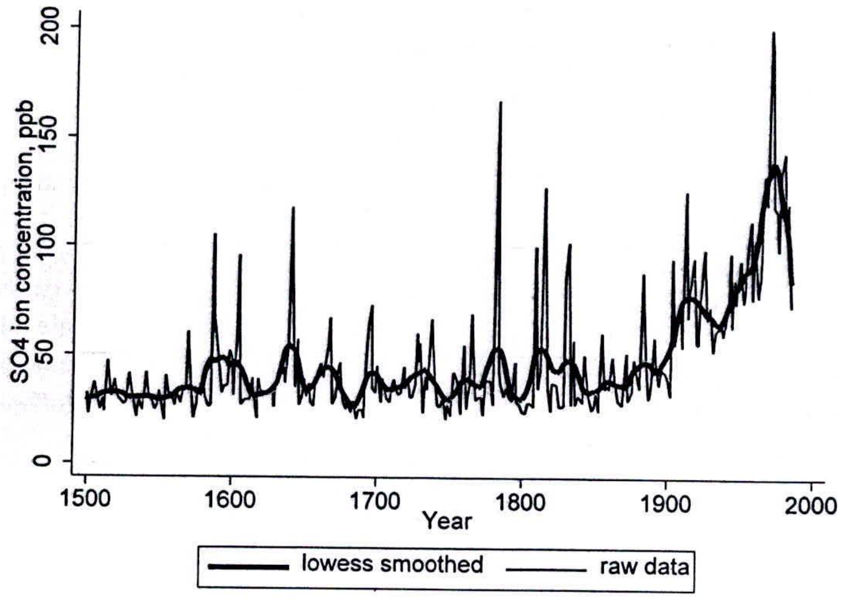
In addition to smoothing scatterplots, **`lowess`** can be used for exploratory time series smoothing. The file *ice.dta* contains results from the Greenland Ice Sheet 2 (GISP2) project described in Mayewski, Holdsworth, and colleagues (1993) and Mayewski, Meeker, and colleagues (1993). Researchers extracted and chemically analyzed an ice core representing more than 100,000 years of climate history. *ice.dta* includes a small fraction of this information: measured non-sea salt sulfate concentration and an index of "Polar Circulation Intensity" since AD 1500.

```
Contains data from C:\data\ice.dta
  obs:           271                          Greenland ice (Mayewski 1995)
 vars:             3                          14 Jul 2005 14:57
 size:         5,962 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   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 8.4 graphs the results. The smoothed curve has been drawn with "thick" width, to visually distinguish it from the raw data. (Type **help linewidthstyle** for other choices of line width.)

```
. graph twoway lowess sulfate year, bwidth(.05) clwidth(thick)
      ||  line sulfate year, clpattern(solid)
      ||  , ytitle("SO4 ion concentration, ppb")
      legend(label(1 "lowess smoothed") label(2 "raw data"))
```

Figure 8.4



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 8.4 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).

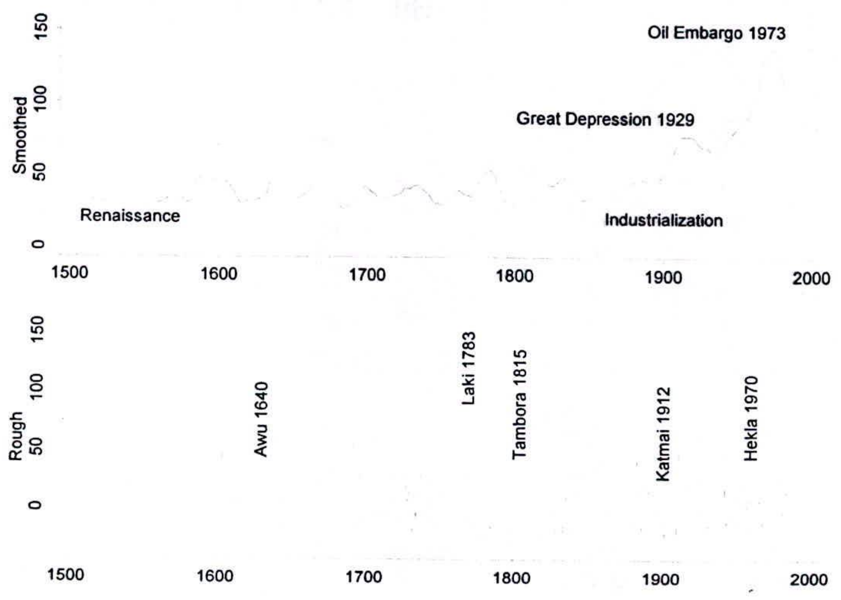
After smoothing time series data, it is often useful to study the smooth and rough (residual) series separately. The following commands create two new variables: lowess-smoothed values of sulfate (*smooth*) and the residuals or rough values (*rough*) calculated by subtracting the smoothed values from 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 8.5 compares the *smooth* and *rough* time series in a pair of graphs annotated using the `text( )` option, then combined.

```
. graph twoway line smooth year, ylabel(0(50)150)  xtitle("")
     ytitle("Smoothed") text(20 1540 "Renaissance")
     text(20 1900 "Industrialization")
     text(90 1860 "Great Depression 1929")
     text(150 1935 "Oil Embargo 1973") saving(fig08_05a, replace)
. graph twoway line rough year, ylabel(0(50)150) xtitle("")
     ytitle("Rough") text(75 1630 "Awu 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_05b, replace)
. graph combine fig08_05a.gph fig08_05b.gph, rows(2)
```

Figure 8.5



Regression with Transformed Variables — 1

By subjecting one or more variables to nonlinear transformation, and then including the transformed variable(s) in a linear regression, we implicitly fit a curvilinear model to the underlying data. Chapters 6 and 7 gave one example of this approach, polynomial regression, which incorporates second (and perhaps higher) powers of at least one x variable among the predictors. Logarithms also are used routinely in many fields. Other common transformations include those of the ladder of powers and Box–Cox transformations, introduced in Chapter 4.

Dataset *tornado.dta* provides a simple illustration involving U.S. tornados from 1916 to 1986 (from the Council on Environmental Quality, 1988).

```
Contains data from C:\data\tornado.dta
  obs:            71                          U.S. tornados 1916-1986
                                                (Council on Env. Quality 1988)
 vars:             4                          16 Jul 2005 14:57
 size:           994 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
year            int    %8.0g                  Year
tornado         int    %8.0g                  Number of tornados
lives           int    %8.0g                  Number of lives lost
avlost          float  %9.0g                  Average lives lost/tornado
-------------------------------------------------------------------------------
Sorted by:  year
```

The number of fatalities decreased over this period, while the number of recognized tornados increased, because of improvements in warnings and our ability to detect more tornados, even those that do little damage. Consequently, the average lives lost per tornado (*avlost*) declined with time, but a linear regression (Figure 8.6, following page) does not well describe this trend. The scatter descends more steeply than the regression line at first, then levels off in the mid-1950s. The regression line actually predicts negative numbers of deaths in later years. Furthermore, average tornado deaths exhibit more variation in early years than later — evidence of heteroskedasticity.

```
. graph twoway scatter avlost year
      ||  lfit avlost year, clpattern(solid)
      ||  , ytitle("Average number of lives lost") xlabel(1920(10)1990)
       xtitle("") legend(off) ylabel(0(1)7) yline(0)
```

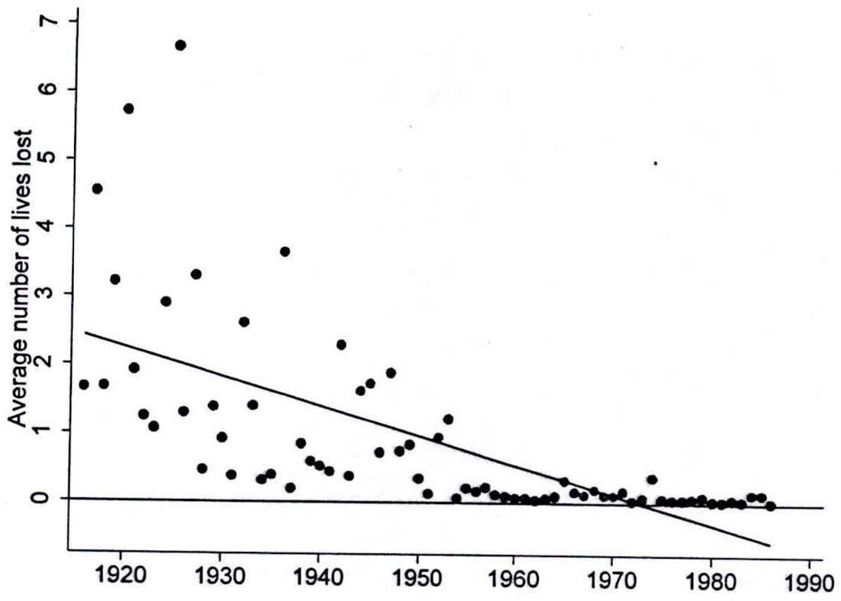


Figure 8.6

The relationship becomes linear, and heteroskedasticity vanishes if we work instead with logarithms of the average number of lives lost (Figure 8.7):

```
. generate loglost = ln(avlost)
. label variable loglost "ln(avlost)"
. regress loglost year
```

```
      Source |       SS       df       MS              Number of obs =      71
-------------+------------------------------           F(  1,    69) =  182.24
       Model |  115.895325     1  115.895325           Prob > F      =  0.0000
    Residual |  43.8807356    69  .63595269            R-squared     =  0.7254
-------------+------------------------------           Adj R-squared =  0.7214
       Total |   159.77606    70  2.28251515           Root MSE      =  .79747

------------------------------------------------------------------------------
     loglost |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        year |  -.0623418    .004618   -13.50   0.000    -.0715545    -.053129
       _cons |   120.5645   9.010312    13.38   0.000     102.5894    138.5395
------------------------------------------------------------------------------
```

```
. predict yhat2
(option xb assumed; fitted values)
. label variable yhat2 "ln(avlost) = 120.56 - .06year"
. label variable loglost "ln(avlost)"
. graph twoway scatter loglost year
        ||  mspline yhat2 year, clpattern(solid) bands(50)
        ||  , ytitle("Natural log(average lives lost)")
        xlabel(1920(10)1990) xtitle("") legend(off) ylabel(-4(1)2)
        yline(0)
```

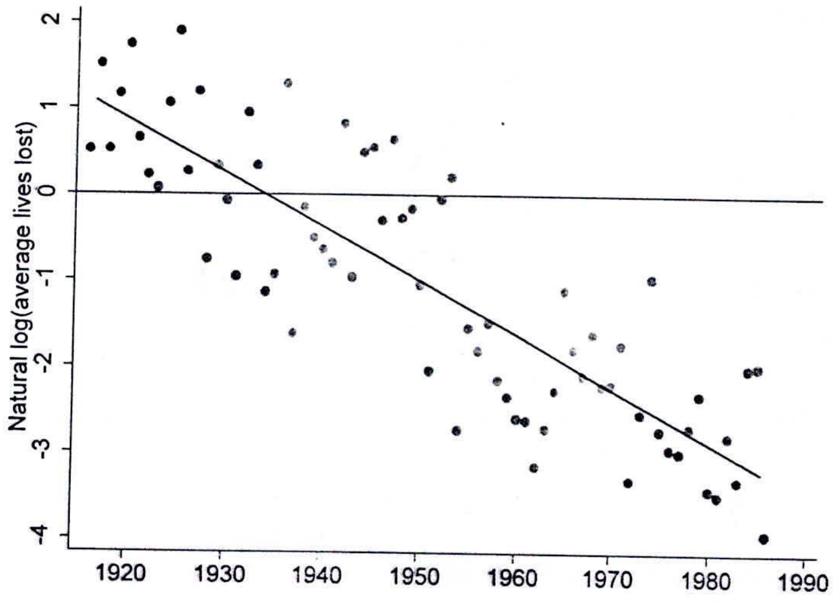


Figure 8.7

The regression model is approximately

predicted ln(*avlost*) = 120.56 – .06*year*

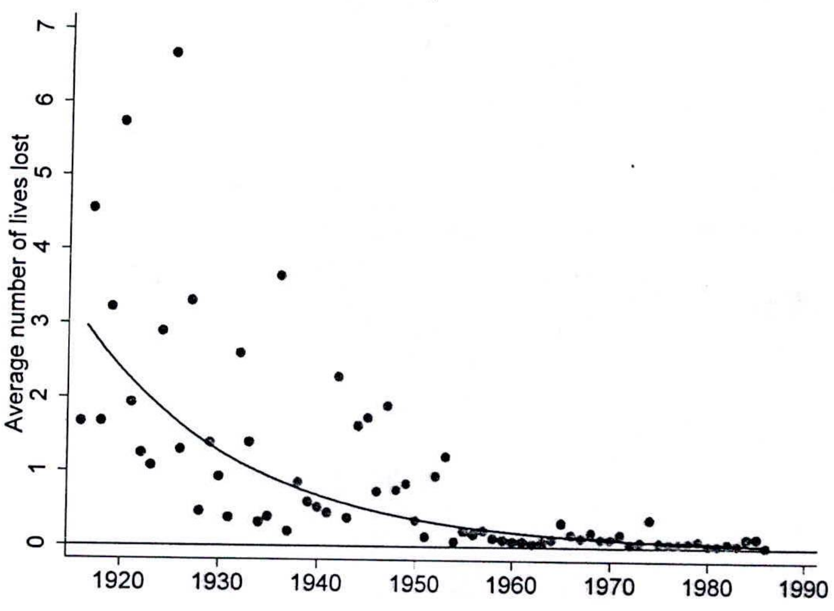
Because we regressed logarithms of lives lost on *year*, the model's predicted values are also measured in logarithmic units. Return these predicted values to their natural units (lives lost) by inverse transformation. in this case exponentiating (*e* to power) *yhat2*:

```
. replace yhat2 = exp(yhat2)
(71 real changes made)
```

Graphing these inverse-transformed predicted values reveals the curvilinear regression model, which we obtained by linear regression with a transformed *y* variable (Figure 8.8). Contrast Figures 8.7 and 8.8 with Figure 8.6 to see how transformation made the analysis both simpler and more realistic.

```
. graph twoway scatter avlost year
      ||  mspline yhat2 year, clpattern(solid) bands(50)
      ||  , ytitle("Average number of lives lost") xlabel(1920(10)1990)
       xtitle("") legend(off) ylabel(0(1)7) yline(0)
```

Figure 8.8



The **boxcox** command employs maximum-likelihood methods to fit curvilinear models involving Box–Cox transformations (introduced in Chapter 4). Fitting a model with Box–Cox transformation of the dependent variable (**model(lhs)** specifies left-hand side) to the tornado data, we obtain results quite similar to the model of Figures 8.7 and 8.8. The **nolog** option in the following command does not affect the model, but suppresses display of log likelihood after each iteration of the fitting process.

```
. boxcox avlost year, model(lhs) nolog

                                                  Number of obs   =         71
                                                  LR chi2(1)      =      92.28
Log likelihood = -7.7185533                       Prob > chi2     =      0.000

------------------------------------------------------------------------------
      avlost |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      /theta |  -.0560959   .0646726    -0.87   0.386    -.1828519      .07066
------------------------------------------------------------------------------

Estimates of scale-variant parameters
----------------------------
             |      Coef.
-------------+--------------
Notrans      |
        year |  -.0661891
       _cons |   127.9713
-------------+--------------
      /sigma |   .8301177
----------------------------
```

```
---------------------------------------------------------
  Test         Restricted      LR statistic     P-Value
   H0:       log likelihood        chi2       Prob > chi2
---------------------------------------------------------
theta = -1     -84.928791         154.42         0.000
theta =  0     -8.0941678           0.75         0.386
theta =  1     -101.50385         187.57         0.000
---------------------------------------------------------
```

The **boxcox** output shows theta = −.056 as the optimal Box–Cox parameter for transforming *avlost*, in order to linearize its relationship with *year*. Therefore, the left-hand-side transformation is

$$alvlost^{(-.056)} = (alvlost^{-.056} - 1)/-.056$$

Box–Cox transformation by a parameter close to zero, such as −.056, produces results similar to the natural-logarithm transformation we applied earlier to this variable "by hand." It is therefore not surprising that the **boxcox** regression model

$$\text{predicted } alvlost^{(-.056)} = 127.97 - .07year$$

resembles the earlier model (predicted ln(*avlost*) = 120.56 – .06*year*) drawn in Figures 8.7 and 8.8. The **boxcox** procedure assumes normal, independent, and identically distributed errors. It does not select transformations with the aim of normalizing residuals, however.

boxcox can fit several types of models, including multiple regressions in which some or all of the right-hand-side variables are transformed by a parameter different from the *y*-variable transformation. It cannot apply different transformations to each separate right-hand-side predictor. To do that, we return to a "by hand" curvilinear-regression approach, as illustrated in the next section.

Regression with Transformed Variables — 2

For a multiple-regression example, we will use data on living conditions in 109 countries found in dataset *nations.dta* (from World Bank 1987; World Resources Institute 1993).

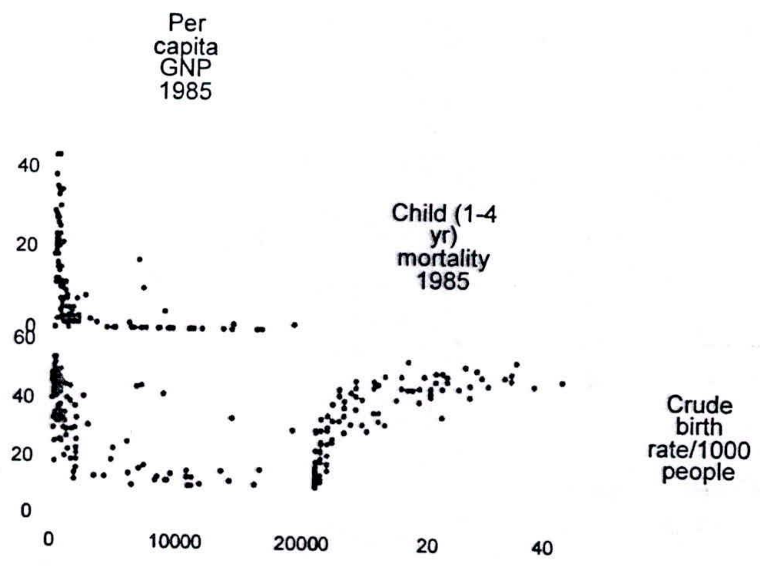
```
Contains data from C:\data\nations.dta
  obs:           109                          Data on 109 nations, ca. 1985
 vars:            15                          16 Jul 2005 14:57
 size:         4,033 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
country         str8   %9s                    Country
pop             float  %9.0g                  1985 population in millions
birth           byte   %8.0g                  Crude birth rate/1000 people
death           byte   %8.0g                  Crude death rate/1000 people
chldmort        byte   %8.0g                  Child (1-4 yr) mortality 1985
infmort         int    %8.0g                  Infant (<1 yr) mortality 1985
life            byte   %8.0g                  Life expectancy at birth 1985
food            int    %8.0g                  Per capita daily calories 1985
energy          int    %8.0g                  Per cap energy consumed, kg oil
gnpcap          int    %8.0g                  Per capita GNP 1985
gnpgro          float  %9.0g                  Annual GNP growth % 65-85
urban           byte   %8.0g                  % population urban 1985
```

```
school1          int     %8.0g                Primary enrollment % age-group
school2          byte    %8.0g                Secondary enroll % age-group
school3          byte    %8.0g                Higher ed. enroll % age-group
-------------------------------------------------------------------------------
```

Relationships among birth rate, per capita gross national product (GNP), and child mortality are not linear, as can be seen clearly in the scatterplot matrix of Figure 8.9. The skewed *gnpcap* and *chldmort* distributions also present potential leverage and influence problems.

```
. graph matrix gnpcap chldmort birth, half
```

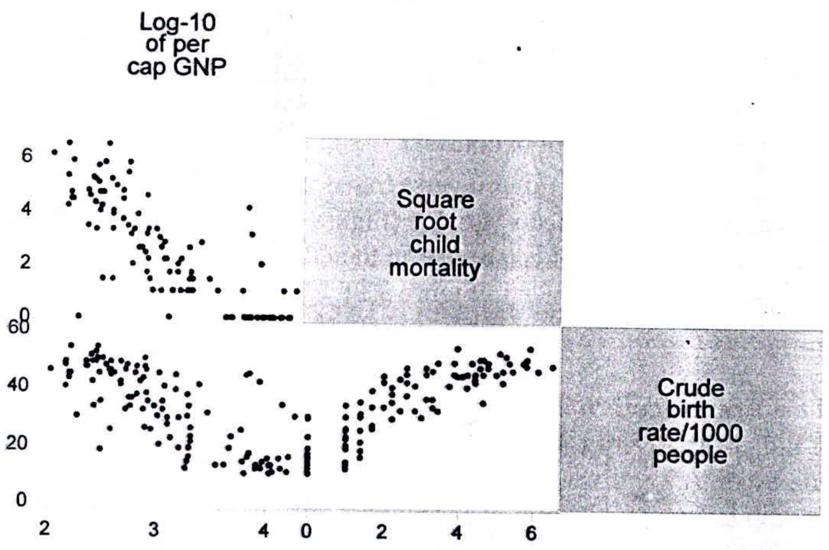
Figure 8.9



Experimenting with ladder-of-powers transformations reveals that the log of *gnpcap* and the square root of *chldmort* have distributions more symmetrical, with fewer outliers or potential leverage points, than the raw variables. More importantly, these transformations largely eliminate the nonlinearities: compare the raw-data scatterplots in Figure 8.9 with their transformed-variables counterparts in Figure 8.10, on the following page,

```
. generate loggnp = log10(gnpcap)
. label variable loggnp "Log-10 of per cap GNP"
. generate srmort = sqrt(chldmort)
. label variable srmort "Square root child mortality"
. graph matrix loggnp srmort birth, half
```

Figure 8.10



We can now apply linear regression using the transformed variables:

```
. regress birth loggnp srmort

      Source |       SS       df       MS                  Number of obs =     109
-------------+------------------------------               F(  2,   106) =  198.06
       Model |  15837.9603     2  7918.98016               Prob > F      =  0.0000
    Residual |  4238.18646   106  39.9828911               R-squared     =  0.7889
-------------+------------------------------               Adj R-squared =  0.7849
       Total |  20076.1468   108  185.890248               Root MSE      =  6.3232

------------------------------------------------------------------------------
       birth |      Coef.   Std. Err.       t     P>|t|      [95% Conf. Interval]
-------------+----------------------------------------------------------------
      loggnp |  -2.353738   1.686255     -1.40    0.166     -5.696903    .9894259
      srmort |   5.577359    .533567     10.45    0.000      4.51951     6.635207
       _cons |   26.19488   6.362687      4.12    0.000     13.58024    38.80953
------------------------------------------------------------------------------
```

Unlike the raw-data regression (not shown), this transformed-variables version finds that per capita gross national product does not significantly affect birth rate once we control for child mortality. The transformed-variables regression fits slightly better: $R^2_a = .7849$ instead of .6715. (We can compare R^2_a across models here only because both have the same untransformed y variable.) Leverage plots would confirm that transformations have much reduced the curvilinearity of the raw-data regression.

Conditional Effect Plots

Conditional effect plots trace the predicted values of *y* as a function of one *x* variable, with other *x* variables held constant at arbitrary values such as their means, medians, quartiles, or extremes. Such plots help with interpreting results from transformed-variables regression.

Continuing with the previous example, we can calculate predicted birth rates as a function of *loggnp*, with *srmort* held at its mean (2.49):

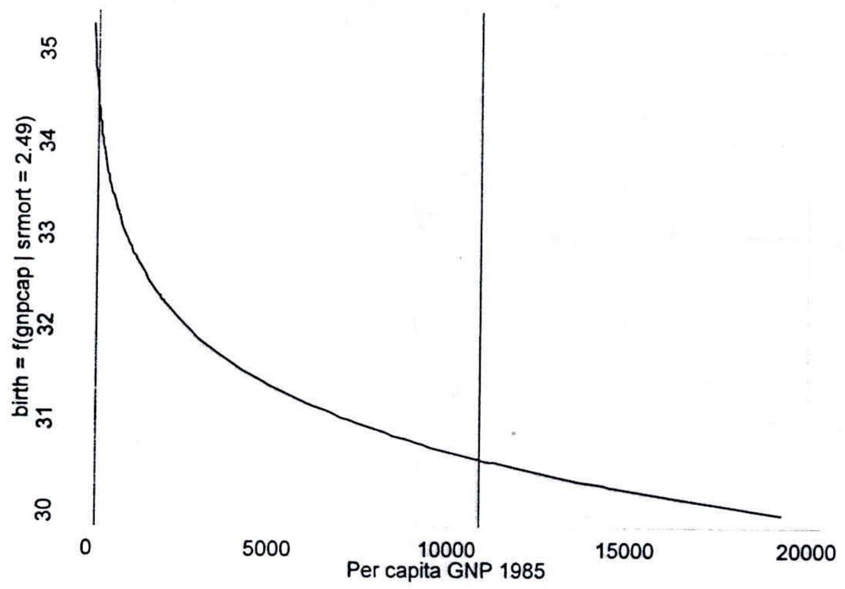
```
. generate yhat1 = _b[_cons] + _b[loggnp]*loggnp + _b[srmort]*2.49
. label variable yhat1 "birth = f(gnpcap | srmort = 2.49)
```

The _b[*varname*] terms refer to the regression coefficient on *varname* from this session's most recent regression. _b[_cons] is the *y*-intercept or constant.

For a conditional effect plot, graph *yhat1* (after inverse transformation if needed, although it is not needed here) against the untransformed *x* variable (Figure 8.11). Because conditional effect plots do not show the scatter of data, it can be useful to add reference lines such as the *x* variable's 10th and 90th percentiles, as shown in Figure 8.11.

```
. graph twoway line yhat1 gnpcap, sort xlabel(,grid) xline(230 10890)
```

Figure 8.11



Similarly, Figure 8.12 depicts predicted birth rates as a function of *srmort*, with *loggnp* held at its mean (3.09):

```
. generate yhat2 = _b[_cons] + _b[loggnp]*3.09 + _b[srmort]*srmort
. label variable yhat2 "birth = f(chldmort | loggnp = 3.09)"
. graph twoway line yhat2 chldmort, sort xlabel(,grid) xline(0 27)
```

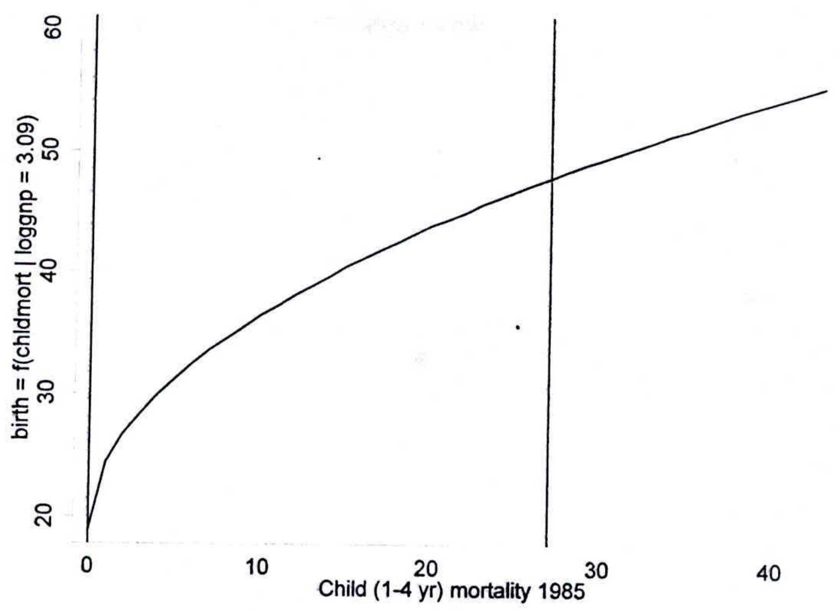


Figure 8.12

How can we compare the strength of different *x* variables' effects? Standardized regression coefficients (beta weights) are sometimes used for this purpose, but they imply a specialized definition of "strength" and can easily be misleading. A more substantively meaningful comparison might come from looking at conditional effect plots drawn with identical *y* scales. This can be accomplished easily by using **graph combine**, and specifying common *y*-axis scales, as done in Figure 8.13. The vertical distances traveled by the predicted values curve, particularly over the middle 80% of the *x* values (between 10th and 90th percentile lines), provide a visual comparison of effect magnitude.

```
. graph combine fig08_11.gph fig08_12.gph, ycommon cols(2) scale(1.25)
```

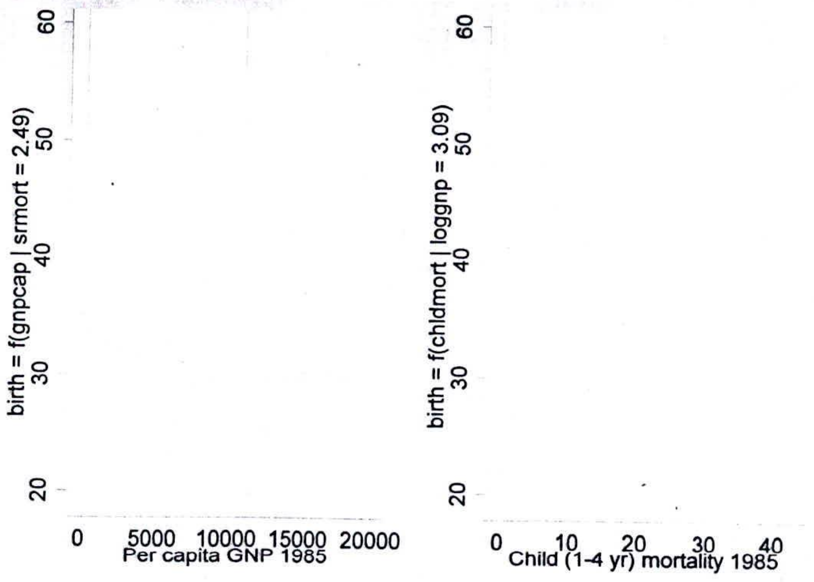


Figure 8.13

Combining several conditional effects plots into one image with common vertical scales, as done in Figure 8.13, allows quick visual comparison of the strength of different effects. Figure 8.13 makes obvious how much stronger is the effect of child mortality on birth rates — as separate plots (Figures 8.11 and 8.12) did not.

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 introduces it using a dataset of simple examples, *nonlin.dta*:

```
Contains data from C:\data\nonlin.dta
  obs:            100                          Nonlinear model examples
                                                 (artificial data)
 vars:              5                          16 Jul 2005 14:57
 size:          2,100  (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
x               byte   %9.0g                  Independent variable
y1              float  %9.0g                  y1 = 10 * 1.03^x + e
y2              float  %9.0g                  y2 = 10 * (1 - .95^x) + e
y3              float  %9.0g                  y3 = 5 + 25/(1+exp(-.1*(x-50)))
                                                + e
y4              float  %9.0g                  y4 = 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 $y1 = 10 \times 1.03^x$. Estimating these parameters from the data, **nl** obtains $y1 = 11.20 \times 1.03^x$, which is reasonably close to the true model.

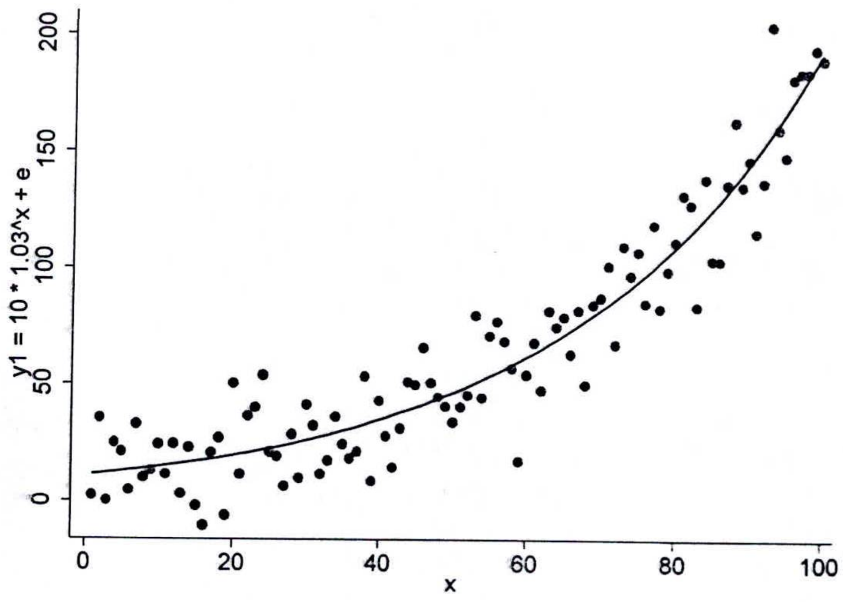
```
. nl exp2 y1 x

(obs = 100)

Iteration 0:  residual SS =  27625.96
Iteration 1:  residual SS =  26547.42
Iteration 2:  residual SS =   26138.3
Iteration 3:  residual SS =  26138.29

      Source |       SS       df       MS            Number of obs =       100
-------------+------------------------------         F(  2,     98) =   1250.42
       Model |  667018.255     2  333509.128         Prob > F      =    0.0000
    Residual |  26138.2933    98  266.717278         R-squared     =    0.9623
-------------+------------------------------         Adj R-squared =    0.9615
       Total |  693156.549   100  6931.56549         Root MSE      =  16.33148
                                                     Res. dev.     =  840.3864
```

```
2-param. exp. growth curve, y1=b1*b2^x
------------------------------------------------------------------------------
          y1 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          b1 |   11.20416   1.146682      9.77   0.000     8.928602    13.47971
          b2 |   1.028838   .0012404    829.41   0.000     1.026376    1.031299
------------------------------------------------------------------------------
  (SE's, P values, CI's, and correlations are asymptotic approximations)
```

The **predict** command obtains predicted values and residuals for a nonlinear model estimated by **nl** . Figure 8.14 graphs predicted values from the previous example, showing the close fit ($R^2 = .96$) between model and data.

```
. predict yhat1
(option yhat assumed; fitted values)

. graph twoway scatter y1 x
       ||  line yhat1 x, sort
       ||  , legend(off) ytitle("y1 = 10 * 1.03^x + e") xtitle("x")
```

Figure 8.14



The **exp2** part of our **nl exp2** ***y1 x*** command specified a particular exponential growth function by calling a brief program named *nlexp2.ado*. Stata includes several such programs, defining 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:
$y = b_1 \exp(-\exp(-b_2(x - b_3)))$

nonlin.dta contains examples corresponding to **exp2** (*y1*), **exp2a** (*y2*), **log4** (*y3*), and **gom4** (*y4*) functions. Figure 8.15 shows curves fit by **nl** to *y2*, *y3*, and *y4*.

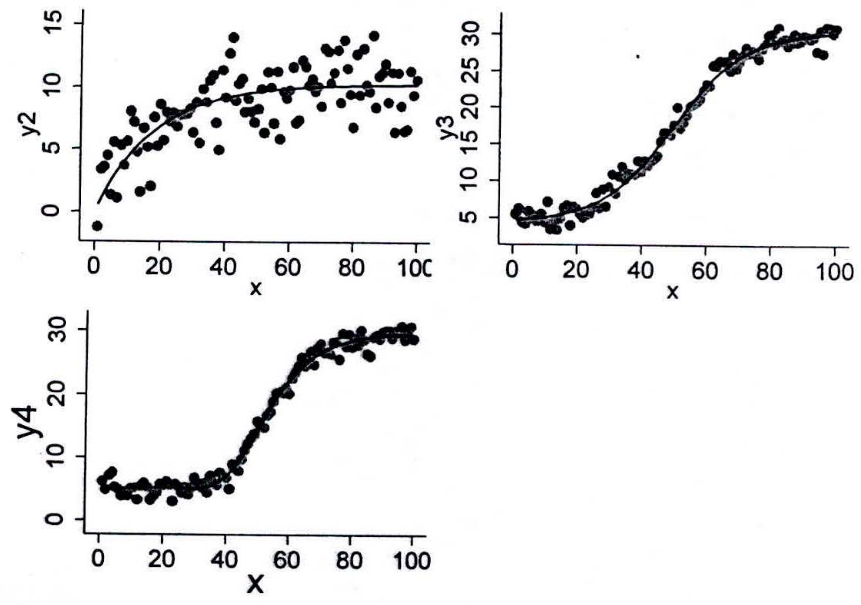


Figure 8.15

Users can write further nl*function* programs of their own. Here is the code for the `nlexp2.ado` program defining a 2-parameter exponential growth model:

```
*! version 1.1.3  12jun1998
program define nlexp2
    version 6
    if "`1'"=="?" {
        global S_2 "2-param. exp. growth curve, $S_E_depv=b1*b2^`2'"
        global S_1 "b1 b2"
/*
    Approximate initial values by regression of log Y on X.
*/
        local exp "[`e(wtype)' `e(wexp)']"
        tempvar Y
        quietly {
            gen `Y' = log(`e(depvar)') if e(sample)
            reg `Y' `2' `exp' if e(sample)
        }
        global b1 = exp(_b[_cons])
        global b2 = exp(_b[`2'])
        exit
    }
    replace `1'=$b1*($b2)^`2'
end
```

This program finds some approximate initial values of the parameters to be estimated, storing these as "global macros" named `b1` and `b2`. It then calculates an initial set of predicted values, as a "local macro" named `1`, employing the initial parameter estimates and the model equation:

```
   replace `1' = $b1 * ($b2)^`2'
```

Subsequent iterations of **nl** will return to this line, calculating new predicted values (replacing the contents of macro `1`) as they refine the parameter estimates `b1` and `b2`. In Stata programs, the notation `$b1` means "the contents of global macro `b1`." Similarly, the notation `` `1' `` means "the contents of local macro `1`."

Before attempting to write your own nonlinear function, examine `nllog4.ado`, `nlgom4.ado`, and others as examples, and consult the manual or **help nl** for explanations. Chapter 14 contains further discussion of macros and other aspects of Stata programming.

Nonlinear Regression — 2

Our second example involves real data, and illustrates some steps that can help in research. Dataset *lichen.dta* concerns measurements of lichen growth observed on the Norwegian arctic island of Svalbard (from Werner 1990). These slow-growing symbionts are often used to date rock monuments and other deposits, so their growth rates interest scientists in several fields.

```
Contains data from C:\data\lichen.dta
  obs:            11                          Lichen growth (Werner 1990)
 vars:             8                          14 Jul 2005 14:57
 size:           572 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
locale          str31  %31s                   Locality and feature
point           str1   %9s                    Control point
date            int    %8.0g                  Date
age             int    %8.0g                  Age in years
rshort          float  %9.0g                  Rhizocarpon short axis mm
rlong           float  %9.0g                  Rhizocarpon long axis mm
pshort          int    %8.0g                  P.minuscula short axis mm
plong           int    %8.0g                  P.minuscula long axis mm
-------------------------------------------------------------------------------
Sorted by:
```

Lichens characteristically exhibit a period of relatively fast early growth, gradually slowing, as suggested by the lowess-smoothed curve in Figure 8.16.

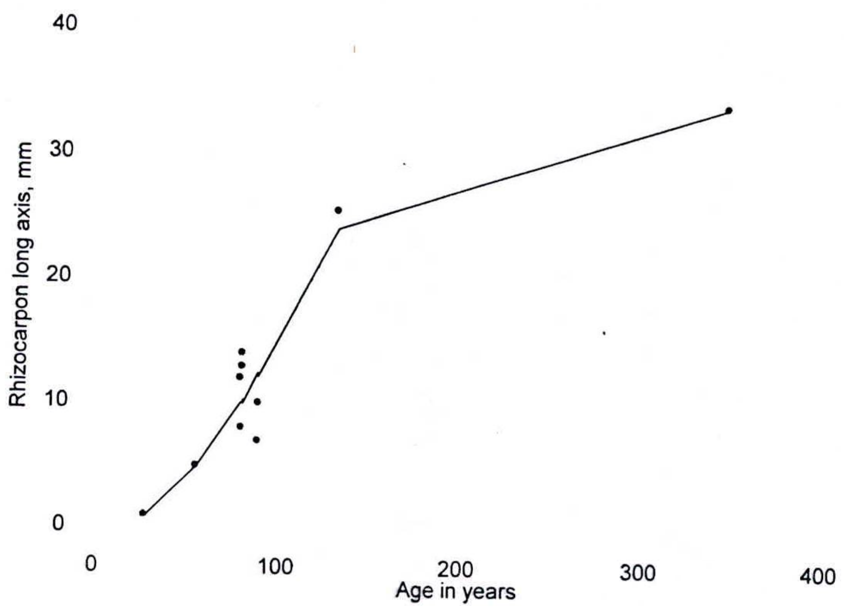


Figure 8.16

Lichenometricians seek to summarize and compare such patterns by drawing growth curves. Their growth curves might not employ an explicit mathematical model, but we can fit one here to illustrate the process of nonlinear regression. Gompertz curves are asymmetrical S-curves, which have been widely used to model biological growth:

$$y = b_1 \exp(-\exp(-b_2(x - b_3)))$$

They might provide a reasonable model for lichen growth.

If we intend to graph a nonlinear model, the data should contain a good range of closely spaced *x* values. Actual ages of the 11 lichen samples in *lichen.dta* range from 28 to 346 years. We can create 89 additional artificial observations, with "ages" from 0 to 352 in 4-year increments, by the following commands:

```
. range newage 0 396 100
obs was 11, now 100

. replace age = newage[_n-11] if age >= .
(89 real changes made)
```

The first command created a new variable, *newage*, with 100 values ranging from 0 to 396 in 4-year increments. In so doing, we also created 89 new artificial observations, with missing values on all variables except *newage*. The **replace** command substitutes the missing artificial-case *age* values with *newage* values, starting at 0. The first 15 observations in our data now look like this:

```
. list rlong age newage in 1/15

     +----------------------+
     | rlong   age   newage |
     |----------------------|
  1. |     1    28        0 |
  2. |     5    56        4 |
  3. |    12    79        8 |
  4. |    14    80       12 |
  5. |    13      80        16 |
     |-------------------------|
  6. |     8      80        20 |
  7. |     7      89        24 |
  8. |    10      89        28 |
  9. |    34     346        32 |
 10. |    34     346        36 |
     |-------------------------|
 11. |  25.5     131        40 |
 12. |     .       0        44 |
 13. |     .       4        48 |
 14. |     .       8        52 |
 15. |     .      12        56 |
     +-------------------------+
```

```
. summarize rlong age newage

    Variable |     Obs        Mean    Std. Dev.       Min        Max
-------------+-----------------------------------------------------
       rlong |      11    14.86364    11.31391          1         34
         age |     100      170.68    104.7042          0        352
      newage |     100         198     116.046          0        396
```

We now could **drop newage**. Only the original 11 observations have nonmissing *rlong* values, so only they will enter into model estimation. Stata calculates predicted values for any observation with nonmissing x values, however. We can therefore obtain such predictions for both the 11 real observations and the 89 artificial ones, which will allow us to graph the regression curve accurately.

Lichen growth starts with a size close to zero, so we chose the **gom3** Gompertz function rather than **gom4** (which incorporates a nonzero takeoff level, the parameter b_0). Figure 8.16 suggests an asymptotic upper limit somewhere near 34, suggesting that 34 should be a good guess or starting value of the parameter b_1. Estimation of this model is accomplished by

```
. nl gom3 rlong age, init(B1=34) nolog

(obs = 11)

      Source |       SS       df       MS             Number of obs =        11
-------------+------------------------------         F(  3,      8) =    125.68
       Model |  3633.16112     3  1211.05371         Prob > F       =    0.0000
    Residual |  77.0888815     8  9.63611018         R-squared      =    0.9792
-------------+------------------------------         Adj R-squared =    0.9714
       Total |     3710.25    11  337.295455         Root MSE       =  3.104208
                                                      Res. dev.      =  52.63435
3-parameter Gompertz function, rlong=b1*exp(-exp(-b2*(age-b3)))
------------------------------------------------------------------------------
       rlong |      Coef.   Std. Err.      t     P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          b1 |   34.36637   2.267186    15.16   0.000      29.13823    39.59451
          b2 |   .0217685   .0060806     3.58   0.007      .0077465    .0357904
          b3 |   88.79701   5.632545    15.76   0.000      75.80834    101.7857
------------------------------------------------------------------------------
  (SE's, P values, CI's, and correlations are asymptotic approximations)
```

A **nolog** option suppresses displaying a log of iterations with the output. All three parameter estimates differ significantly from 1.

We obtain predicted values using **predict**, and graph these to see the regression curve. The **yline** option is used to display the lower and estimated upper limits (0 and 34.366) of this curve in Figure 8.17.

```
. predict yhat
(option yhat assumed; fitted values)

. graph twoway scatter rlong age
      || mspline yhat age, clpattern(solid) bands(50)
      || , legend(off) yline(0 34.366)
      ytitle("Rhizocarpon long axis, mm") xlabel(0(100)400, grid)
```

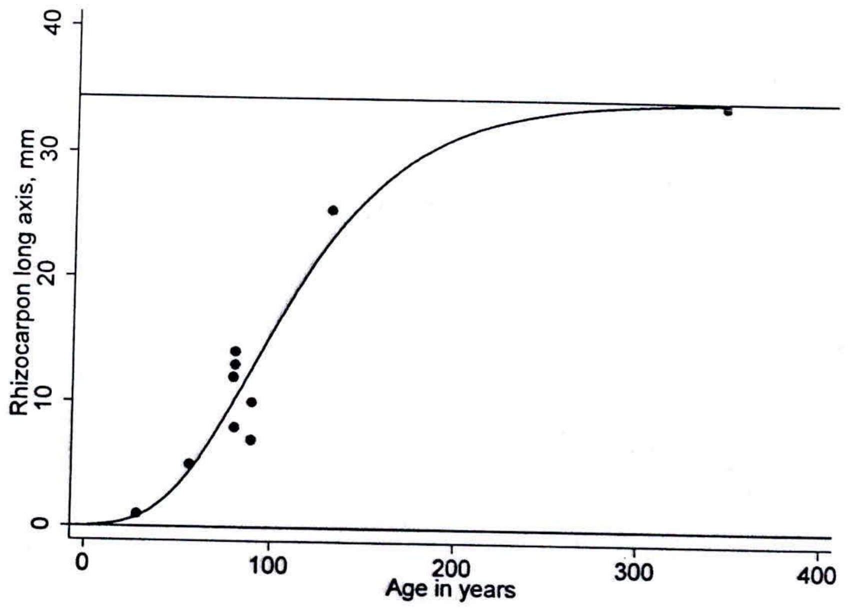


Figure 8.17

Especially when working with sparse data or a relatively complex model, nonlinear regression programs can be quite sensitive to their initial parameter estimates. The **init** option with **nl** permits researchers to suggest their own initial values if the default values supplied by an nl*function* program do not seem to work. Previous experience with similar data, or publications by other researchers, could help supply suitable initial values. Alternatively, we could estimate through trial and error by employing **generate** to calculate predicted values based on arbitrarily-chosen sets of parameter values and **graph** to compare the resulting predictions with the data.

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 methods aim to achieve almost the efficiency of OLS with ideal data and substantially better-than-OLS efficiency in non-ideal (for example, nonnormal errors) situations. "Robust regression" encompasses a variety of different techniques, each with advantages and drawbacks for dealing with problematic data. This chapter introduces two varieties of robust regression, **rreg** and **qreg**, and briefly compares their results with those of OLS (**regress**).

rreg and **qreg** resist the pull of outliers, giving them better-than-OLS efficiency in the face of nonnormal, heavy-tailed error distributions. They share the OLS assumption that errors are independent and identically distributed, however. As a result, their standard errors, tests, and confidence intervals are not trustworthy in the presence of heteroskedasticity or correlated errors. To relax the assumption of independent, identically distributed errors when using **regress** or certain other modeling commands (although not **rreg** or **qreg**), Stata offers options that estimate robust standard errors.

For clarity, this chapter focuses mostly on two-variable examples, but robust multiple regression or *N*-way ANOVA are straightforward using the same commands. Chapter 14 returns to the topic of robustness, showing how we can use Monte Carlo experiments to evaluate competing statistical techniques.

Several of the techniques described in this chapter are available through menu selections:

Statistics – Nonparametric analysis – Quantile regression

Statistics – Linear regression and related – Linear regression – Robust SE

Example Commands

```
. 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 new 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*.

```
. bsqreg y x1 x2 x3, rep(100)
```

Performs quantile regression, with standard errors estimated by bootstrap data resampling with 100 repetitions (default is **rep(20)**).

```
. predict e, resid
```

Calculates residual values (arbitrarily named *e*) after any **regress**, **rreg**, **qreg**, or **bsqreg** command. Similarly, **predict *yhat*** calculates the predicted values of *y*. Other **predict** options apply, with some restrictions.

```
. regress y x1 x2 x3, robust
```

Performs OLS regression of *y* on three predictors. Coefficient variances, and hence standard errors, are estimated by a robust method (Huber/White or sandwich) that does not assume identically distributed errors. With the **cluster()** option, one source of correlation among the errors can be accommodated as well. The *User's Guide* describes the reasoning behind these methods.

Regression with Ideal Data

To clarify the issue of robustness, we will explore the small (n = 20) contrived dataset *robust1.dta*:

```
Contains data from C:\data\robust1.dta
  obs:            20                          Robust regression examples 1
                                                (artificial data)
 vars:            10                          17 Jul 2005 09:35
 size:           880  (99.9% of memory free)
```

```
---------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
---------------------------------------------------------------------------
x               float  %9.0g                  Normal X
e1              float  %9.0g                  Normal errors
y1              float  %9.0g                  y1 = 10 + 2*x + e1
e2              float  %9.0g                  Normal errors with 1 outlier
y2              float  %9.0g                  y2 = 10 + 2*x + e2
x3              float  %9.0g                  Normal X with 1 leverage obs.
e3              float  %9.0g                  Normal errors with 1 extreme
y3              float  %9.0g                  y3 = 10 + 2*x3 + e3
e4              float  %9.0g                  Skewed errors
y4              float  %9.0g                  y4 = 10 + 2*x + e4
---------------------------------------------------------------------------
Sorted by:
```

The variables *x* and *e1* each contain 20 random values from independent standard normal distributions. *y1* contains 20 values produced by the regression model:

$$y1 = 10 + 2x + e1$$

The commands that manufactured these first three variables are

```
. clear
. set obs 20
. generate x = invnorm(uniform())
. generate e1 = invnorm(uniform())
. generate y1 = 10 + 2*x + e1
```

With real data, coding mistakes and measurement errors sometimes create wildly incorrect values. To simulate this, we might shift the second observation's error from –0.89 to 19.89:

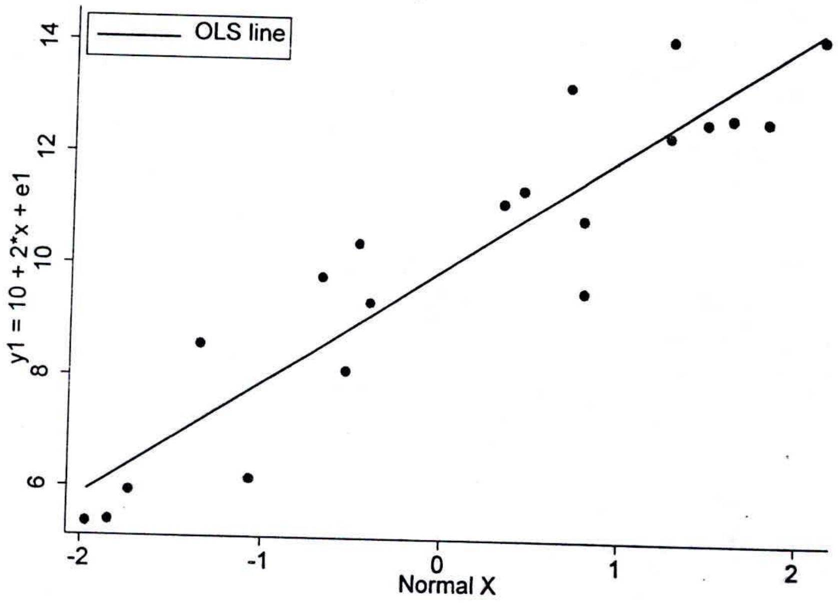
```
. generate e2 = e1
. replace e2 = 19.89 in 2
. generate y2 = 10 + 2*x + e2
```

Similar manipulations produce the other variables in *robust1.dta*.

y1 and *x* present an ideal regression problem: the expected value of *y1* really is a linear function of *x*, and errors come from normal, independent, and identical distributions — because we defined them that way. OLS does a good job of estimating the true intercept (10) and slope (2), obtaining the line shown in Figure 9.1.

```
. regress y1 x

      Source |       SS       df       MS              Number of obs =      20
-------------+------------------------------           F(  1,    18) =  108.25
       Model |  134.059351     1  134.059351           Prob > F      =  0.0000
    Residual |    22.29157    18  1.23842055           R-squared     =  0.8574
-------------+------------------------------           Adj R-squared =  0.8495
       Total |  156.350921    19  8.22899586           Root MSE      =  1.1128

------------------------------------------------------------------------------
          y1 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   2.048057   .1968465    10.40   0.000     1.634498    2.461616
       _cons |   9.963161   .2499861    39.85   0.000      9.43796    10.48836
------------------------------------------------------------------------------

. predict yhat1o
. graph twoway scatter y1 x
      ||  line yhatlo x, clpattern(solid) sort
      ||  , ytitle("y1 = 10 + 2*x + e1") legend(order(2)
         label(2 "OLS line") position(11) ring(0) cols(1))
```

Figure 9.1



An iteratively reweighted least squares (IRLS) procedure, **rreg**, obtains robust regression estimates. 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 y1 x

   Huber iteration 1:  maximum difference in weights = .35774407
   Huber iteration 2:  maximum difference in weights = .02181578
Biweight iteration 3:  maximum difference in weights = .14421371
Biweight iteration 4:  maximum difference in weights = .01320276
Biweight iteration 5:  maximum difference in weights = .00265408

Robust regression estimates                            Number of obs =      20
                                                       F(  1,    18) =   79.96
                                                       Prob > F      =  0.0000

------------------------------------------------------------------------------
          y1 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   2.047813   .2290049     8.94   0.000     1.566692    2.528935
       _cons |   9.936163   .2908259    34.17   0.000     9.325161    10.54717
------------------------------------------------------------------------------
```

This "ideal data" example includes no serious outliers, so here **`rreg`** is unneeded. The **`rreg`** intercept and slope estimates resemble those obtained by **`regress`** (and are not far from the true values 10 and 2), but they have slightly larger estimated standard errors. Given normal i.i.d. errors, as in this example, **`rreg`** theoretically possesses about 95% of the efficiency of OLS.

`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 and provides another method with good resistance to outliers:

```
. qreg y1 x

Iteration  1:  WLS sum of weighted deviations =   17.711531

Iteration  1: sum of abs. weighted deviations =   17.130001
Iteration  2: sum of abs. weighted deviations =   16.858602

Median regression                                    Number of obs =        20
  Raw sum of deviations       46.84 (about 10.4)
  Min sum of deviations     16.8586                  Pseudo R2     =    0.6401

------------------------------------------------------------------------------
      y1 |      Coef.   Std. Err.       t     P>|t|     [95% Conf. Interval]
---------+--------------------------------------------------------------------
       x |   2.139896    .2590447      8.26   0.000      1.595664    2.684129
   _cons |    9.65342    .3564108     27.09   0.000      8.904628    10.40221
------------------------------------------------------------------------------
```

Although **`qreg`** obtains reasonable parameter estimates, its standard errors here exceed those of **`regress`** (OLS) and **`rreg`**. Given ideal data, **`qreg`** is the least efficient of these three estimators. The following sections view their performance with less ideal data.

Y Outliers

The variable *y2* is identical to *y1*, but with one outlier caused by the "wild" error of observation #2. OLS has little resistance to outliers, so this shift in observation #2 (at upper left in Figure 9.2) substantially changes the **`regress`** results:

```
. regress y2 x

  Source |       SS       df       MS                  Number of obs =      20
---------+------------------------------               F(  1,    18) =    0.97
   Model |  18.764271     1   18.764271                Prob > F      =  0.3378
Residual | 348.233471    18  19.3463039                R-squared     =  0.0511
---------+------------------------------               Adj R-squared = -0.0016
   Total | 366.997742    19  19.3156706                Root MSE      =  4.3984

------------------------------------------------------------------------------
      y2 |      Coef.   Std. Err.       t     P>|t|     [95% Conf. Interval]
---------+--------------------------------------------------------------------
       x |   .7662304    .7780232      0.98   0.338     -.8683356    2.400796
   _cons |    11.1579    .9880542     11.29   0.000      9.082078    13.23373
------------------------------------------------------------------------------
```

```
. predict yhat2o
(option xb assumed; fitted values)

. label variable yhat2o "OLS line (regress)"
```

The outlier raises the OLS intercept (from 9.936 to 11.1579) and lessens the slope (from 2.048 to 0.766). R^2 has dropped from .8574 to .0511. Standard errors quadrupled, and the OLS slope (solid line in Figure 9.2) no longer significantly differs from zero.

The outlier has little impact on **rreg**, however, as shown by the dashed line in Figure 9.2. The robust coefficients barely change, and remain close to the true parameters 10 and 2; nor do the robust standard errors increase much.

```
. rreg y2 x, nolog genwt(rweight2)

Robust regression estimates                           Number of obs =      19
                                                      F(  1,    17) =   63.01
                                                      Prob > F      =  0.0000

------------------------------------------------------------------------------
          y2 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   1.979015   .2493146     7.94   0.000     1.453007    2.505023
       _cons |   10.00897   .3071265    32.59   0.000     9.360986    10.65695
------------------------------------------------------------------------------

. predict yhat2r
(option xb assumed; fitted values)

. label variable yhat2r "robust regression (rreg)"

. graph twoway scatter y2 x
      || line yhat2o x, clpattern(solid) sort
      || line yhat2r x, clpattern(longdash) sort
      || , ytitle("y2 = 10 + 2*x + e2")
      legend(order(2 3) position(1) ring(0) cols(1) margin(sides))
```

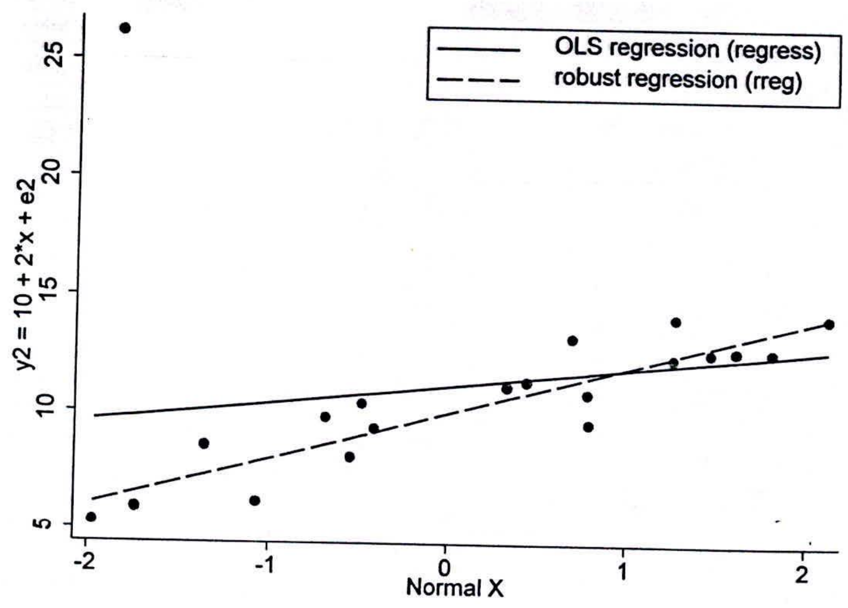


Figure 9.2

The **nolog** option above caused Stata not to print the iteration log. The **genwt(*rweight2*)** option saved robust weights as a variable named *rweight2*.

```
. predict resid2r, resid

. list y2 x resid2r rweight2

     +-----------------------------------------+
     |   y2       x      resid2r     rweight2  |
     |-----------------------------------------|
  1. |  5.37   -1.97   -.7403071    .94644465  |
  2. | 26.19   -1.85    19.84221            .  |
  3. |  5.93   -1.74   -.6354816    .96037073  |
  4. |  8.58   -1.36    1.262494     .8493384  |
  5. |  6.16   -1.07   -1.731421     .7257631  |
     |-----------------------------------------|
  6. |  9.80   -0.69    1.156554    .87273631  |
  7. |  8.12   -0.55   -.8005088    .93758391  |
  8. | 10.40   -0.49     1.36075    .82606386  |
  9. |  9.35   -0.42      .17221    .99712388  |
 10. | 11.16    0.33    .4979581    .97581674  |
     |-----------------------------------------|
 11. | 11.40    0.44    .5202664    .97360863  |
 12. | 13.26    0.69    1.885513    .68048066  |
 13. | 10.88    0.78   -.6725981    .95572833  |
 14. |  9.58    0.79   -1.992389    .64644918  |
 15. | 12.41    1.26   -.0925257    .99913568  |
     |-----------------------------------------|
 16. | 14.14    1.27    1.617655    .75887073  |
 17. | 12.66    1.47   -.2581189    .99338589  |
 18. | 12.74    1.61   -.4551811    .97957817  |
 19. | 12.70    1.81   -.8909839    .92307041  |
 20. | 14.19    2.12   -.0144787    .99997651  |
     +-----------------------------------------+
```

Residuals near zero produce weights near one; farther-out residuals get progressively lower weights. Observation #2 has been automatically set aside as too influential because of Cook's $D > 1$. **rreg** assigns its *rweight2* as "missing," so this observation has no effect on the final estimates. The same final estimates, although not the correct standard errors or tests, could be obtained using **regress** with analytical weights (results not shown):

```
. regress y2 x [aweight = rweight2]
```

Applied to the regression of *y2* on *x*, **qreg** also resists the outlier's influence and performs better than **regress**, but not as well as **rreg**. **qreg** appears less efficient than **rreg**, and in this sample its coefficient estimates are slightly farther from the true values of 10 and 2.

```
. qreg y2 x, nolog

Median regression                                    Number of obs =        20
  Raw sum of deviations     56.68 (about 10.88)
  Min sum of deviations 36.20036                     Pseudo R2     =    0.3613

------------------------------------------------------------------------------
          y2 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   1.821428   .4105944     4.44   0.000     .9588014    2.684055
       _cons |     10.115   .5088526    19.88   0.000     9.045941    11.18406
------------------------------------------------------------------------------
```

Monte Carlo researchers have also noticed that the standard errors calculated by **qreg** sometimes underestimate the true sample-to-sample variation, particularly with smaller samples. As an alternative, Stata provides the command **bsqreg**, which performs the same median or quantile regression as **qreg** , but employs bootstrapping (data resampling) to estimate the standard errors. The option **rep( )** controls the number of repetitions. Its default is **rep(20)**, which is enough for exploratory work. Before reaching "final" conclusions, we might take the time to draw 200 or more bootstrap samples. Both **qreg** and **bsqreg** fit identical models. In the example below, **bsqreg** also obtains similar standard errors. Chapter 14 returns to the topic of bootstrapping.

```
. bsqreg y2 x, rep(50)

(fitting base model)
(bootstrapping ..................................................)

Median regression, bootstrap(50) SEs                Number of obs =        20
  Raw sum of deviations    56.68 (about 10.88)
  Min sum of deviations 36.20036                    Pseudo R2     =    0.3613

------------------------------------------------------------------------------
          y2 |      Coef.   Std. Err.       t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   1.821428   .4084728     4.46   0.000      .9632587    2.679598
       _cons |     10.115   .4774718    21.18   0.000      9.111869    11.11813
------------------------------------------------------------------------------
```

X Outliers (Leverage)

rreg, **qreg**, and **bsqreg** deal comfortably with *y*-outliers, unless the observations with unusual *y* values have unusual *x* values (leverage) too. The *y3* and *x3* variables in *robust.dta* present an extreme example of leverage. Apart from the leverage observation (#2), these variables equal *y1* and *x*.

The high leverage of observation #2, combined with its exceptional *y3* value, make it influential: **regress** and **qreg** both track this outlier, reporting that the "best-fitting" line has a negative slope (Figure 9.3).

```
. regress y3 x3

      Source |       SS       df       MS                  Number of obs =      20
-------------+------------------------------               F(  1,    18) =   11.01
       Model |  139.306724     1  139.306724               Prob > F      =  0.0038
    Residual |  227.691018    18   12.649501               R-squared     =  0.3796
-------------+------------------------------               Adj R-squared =  0.3451
       Total |  366.997742    19  19.3156706               Root MSE      =  3.5566

------------------------------------------------------------------------------
          y3 |      Coef.   Std. Err.       t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x3 |  -.6212248   .1871973    -3.32   0.004     -1.014512    -.227938
       _cons |   10.80931   .8063436    13.41   0.000      9.115244    12.50337
------------------------------------------------------------------------------

. predict yhat3o

. label variable yhat3o "OLS regression (regress)"
. qreg y3 x3, nolog

Median regression                                   Number of obs =        20
  Raw sum of deviations     56.68 (about 10.88)
  Min sum of deviations 56.19466                    Pseudo R2     =    0.0086

------------------------------------------------------------------------------
          y3 |      Coef.   Std. Err.       t     P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x3 |  -.6222217    .347103     -1.79    0.090     -1.351458    .1070146
       _cons |   11.36533   1.419214      8.01    0.000      8.383676    14.34699
------------------------------------------------------------------------------

. predict yhat3q

. label variable yhat3q "median regression (qreg)"

. rreg y3 x3, nolog
Robust regression estimates                            Number of obs =      19
                                                       F(  1,     17) =   63.01
                                                       Prob > F       =  0.0000

------------------------------------------------------------------------------
          y3 |      Coef.   Std. Err.       t     P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x3 |   1.979015   .2493146      7.94    0.000      1.453007    2.505023
       _cons |   10.00897   .3071265     32.59    0.000      9.360986    10.65695
------------------------------------------------------------------------------

. predict yhat3r

. label variable yhat3r "robust regression (rreg)"

. graph twoway scatter y3 x3
       || line yhat3o x3, clpattern(solid) sort
       || line yhat3r x3, clpattern(longdash) sort
       || line yhat3q x3, clpattern(shortdash) sort ,
      ytitle("y3 = 10 + 2*x + e3") legend(order(4 3 2) position(5)
         ring(0) cols(1) margin(sides)) ylabel(-30(10)30)
```

Figure 9.3

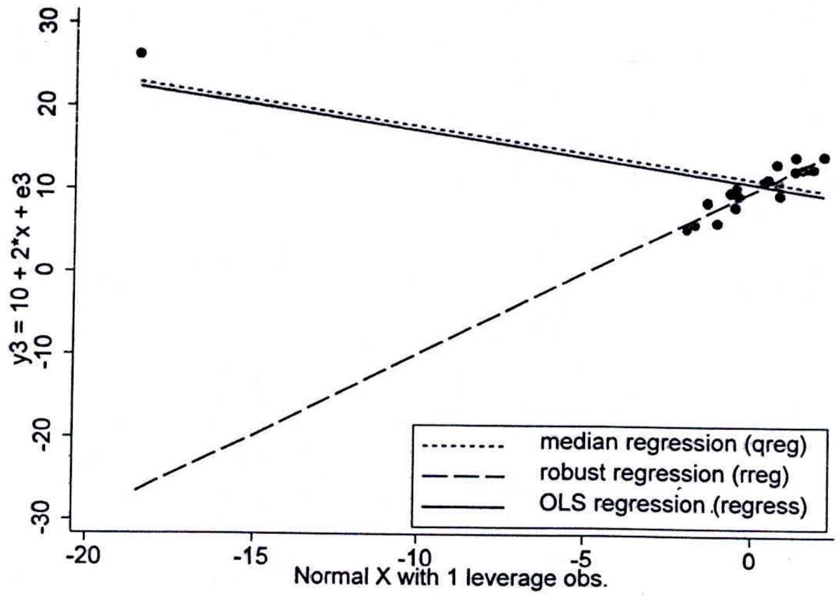


Figure 9.3 illustrates that **regress** and **qreg** are not robust against leverage (*x*-outliers). The **rreg** program, however, not only downweights large-residual observations (which by itself gives little protection against leverage), but also automatically sets aside observations with Cook's *D* (influence) statistics greater than 1. This happened when we regressed *y3* on *x3*; **rreg** ignored the one influential observation and produced a more reasonable regression line with a positive slope, based on the remaining 19 observations.

Setting aside high-influence observations, as done by **rreg**, provides a simple but not foolproof way to deal with leverage. More comprehensive methods, termed bounded-influence regression, also exist and could be implemented in a Stata program.

The examples in Figures 9.2 and 9.3 involve single outliers, but robust procedures can handle more. Too many severe outliers, or a cluster of similar outliers, might cause them to break down. But in such situations, which are often noticeable in diagnostic plots, the analyst must question whether fitting a linear model makes sense. It might be worthwhile to seek an explicit model for what is causing the outliers to be different.

Monte Carlo experiments (illustrated in Chapter 14) confirm that estimators like **rreg** and **qreg** generally remain unbiased, with better-than-OLS efficiency, when applied to heavy-tailed (outlier-prone) but symmetrical error distributions. The next section illustrates what can happen when errors have asymmetrical distributions.

Asymmetrical Error Distributions

The variable *e4* in *robust1.dta* has a skewed and outlier-filled distribution: *e4* equals *e1* (a standard normal variable) raised to the fourth power, and then adjusted to have 0 mean. These skewed errors, plus the linear relationship with *x*, define the variable $y4 = 10 + 2x + e4$. Regardless of an error distribution's shape, OLS remains an unbiased estimator. Over the long run, its estimates should center on the true parameter values.

```
. regress y4 x

      Source |       SS       df       MS                  Number of obs =      20
-------------+------------------------------               F(  1,    18) =    6.97
       Model |  155.870383     1  155.870383               Prob > F      =  0.0166
    Residual |  402.341909    18  22.3523283               R-squared     =  0.2792
-------------+------------------------------               Adj R-squared =  0.2392
       Total |  558.212291    19  29.3795943               Root MSE      =  4.7278

------------------------------------------------------------------------------
          y4 |      Coef.   Std. Err.       t     P>|t|      [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   2.208388    .8362862     2.64    0.017      .4514157     3.96536
       _cons |   9.975681    1.062046     9.39    0.000      7.744406    12.20696
------------------------------------------------------------------------------
```

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, as with *e4*, **rreg** will downweight mostly one side, resulting in noticeably biased *y*-intercept estimates.

```
. rreg y4 x, nolog

Robust regression estimates                      Number of obs =        20
                                                 F(  1,     18) = 1319.29
                                                 Prob > F       =  0.0000

------------------------------------------------------------------------------
          y4 |      Coef.   Std. Err.       t     P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   1.952073   .0537435     36.32    0.000     1.839163    2.064984
       _cons |   7.476669   .0682518    109.55    0.000     7.333278    7.620061
------------------------------------------------------------------------------
```

Although the **rreg** *y*-intercept in Figure 9.4 is too low, the slope remains parallel to the OLS line and the true model. In fact, being less affected by outliers, the **rreg** slope (1.95) is closer to the true slope (2) and has a much smaller standard error than that of **regress**. This illustrates the tradeoff of 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.

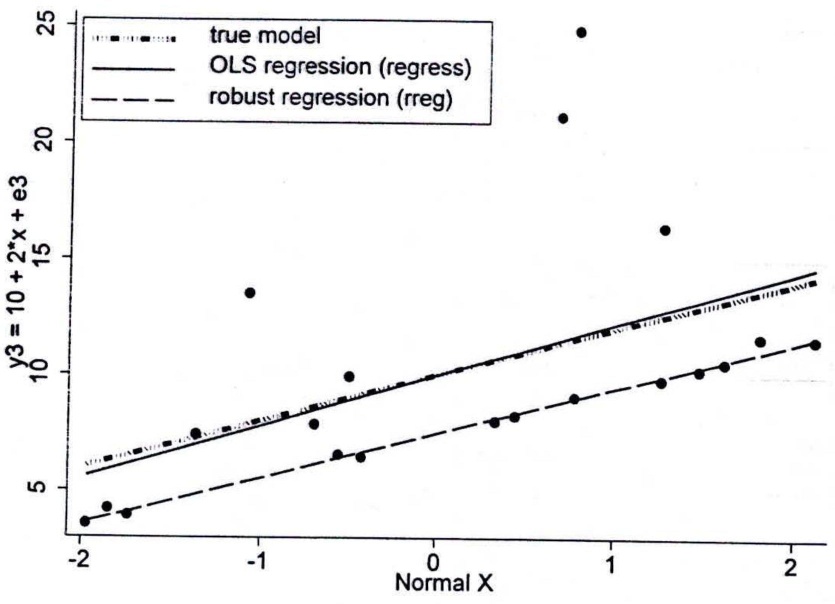


Figure 9.4

Robust Analysis of Variance

rreg can also perform robust analysis of variance or covariance once the model is recast in regression form. For illustration, consider the data on college faculty salaries in *faculty.dta*.

```
Contains data from C:\data\faculty.dta
  obs:           226                           College faculty salaries
 vars:             6                           17 Jul 2005 09:32
 size:         2,938  (99.9% of memory free)
```

```
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
rank            byte   %8.0g       rank       Academic rank
gender          byte   %8.0g       sex        Gender (dummy variable)
female          byte   %8.0g                  Gender (effect coded)
assoc           byte   %8.0g                  Assoc Professor (effect coded)
full            byte   %8.0g                  Full Professor (effect coded)
pay             float  %9.0g                  Annual salary
-------------------------------------------------------------------------------
Sorted by:
```

Faculty salaries increase with rank. In this sample, men have higher average salaries:

```
. table gender rank, contents(mean pay)

----------------------------------------
Gender     |
(dummy     |          Academic rank
variable)  |    Assist       Assoc       Full
-----------+----------------------------
      Male |     29280    38622.22    52084.9
    Female |  28711.04    38019.05      47190
----------------------------------------
```

An ordinary (OLS) analysis of variance indicates that both *rank* and *gender* significantly affect salary. Their interaction is not significant.

```
. anova pay rank gender rank*gender

                          Number of obs =      226     R-squared     =  0.7305
                          Root MSE      =  5108.21     Adj R-squared =  0.7244

                 Source |  Partial SS    df       MS           F     Prob > F
            ------------+----------------------------------------------------
                  Model |  1.5560e+10     5  3.1120e+09     119.26     0.0000
                        |
                   rank |  7.6124e+09     2  3.8062e+09     145.87     0.0000
                 gender |   127361829     1   127361829       4.88     0.0282
            rank*gender |  87997720.1     2  43998860.1       1.69     0.1876
                        |
               Residual |  5.7406e+09   220  26093824.5
            ------------+----------------------------------------------------
                  Total |  2.1300e+10   225  94668810.3
```

But salary is not normally distributed, and the senior-rank averages reflect the influence of a few highly paid outliers. Suppose we want to check these results by performing a robust analysis of variance. We need effect-coded versions of the *rank* and *gender* variables, which this dataset also contains.

```
. tabulate gender female

    Gender |
    (dummy | Gender (effect coded)
 variable) |        -1          1 |     Total
-----------+----------------------+----------
      Male |       149          0 |       149
    Female |         0         77 |        77
-----------+----------------------+----------
     Total |       149         77 |       226
```

```
. tabulate rank assoc

  Academic |   Assoc Professor (effect coded)
      rank |        -1          0          1 |     Total
-----------+---------------------------------+----------
    Assist |        64          0          0 |        64
     Assoc |         0          0        105 |       105
      Full |         0         57          0 |        57
-----------+---------------------------------+----------
     Total |        64         57        105 |       226

. tab rank full

  Academic |   Full Professor (effect coded)
      rank |        -1          0          1 |     Total
-----------+---------------------------------+----------
    Assist |        64          0          0 |        64
     Assoc |         0        105          0 |       105
      Full |         0          0         57 |        57
-----------+---------------------------------+----------
     Total |        64        105         57 |       226
```

If *faculty.dta* did not already have these effect-coded variables (*female*, *assoc*, and *full*), we could create them from *gender* and *rank* using a series of **generate** and **replace** statements. We also need two interaction terms representing female associate professors and female full professors:

```
. generate femassoc = female*assoc
. generate femfull = female*full
```

Males and assistant professors are "omitted categories" in this example. Now we can duplicate the previous ANOVA using regression:

```
. regress pay assoc full female femassoc femfull

  Source |       SS       df       MS                  Number of obs =     226
---------+------------------------------               F(  5,   220) =  119.26
   Model |  1.5560e+10     5  3.1120e+09               Prob > F      =  0.0000
Residual |  5.7406e+09   220  26093824.5               R-squared     =  0.7305
---------+------------------------------               Adj R-squared =  0.7244
   Total |  2.1300e+10   225  94668810.3               Root MSE      =  5108.2

------------------------------------------------------------------------------
     pay |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   assoc |  -663.8995   543.8499     -1.22   0.223      -1735.722    407.9229
    full |   10652.92   783.9227     13.59   0.000       9107.957    12197.88
  female |  -1011.174   457.6938     -2.21   0.028      -1913.199   -109.1483
femassoc |   709.5864   543.8499      1.30   0.193      -362.2359    1781.409
 femfull |  -1436.277   783.9227     -1.83   0.068      -2981.236    108.6819
   _cons |   38984.53   457.6938     85.18   0.000       38082.51    39886.56
------------------------------------------------------------------------------

. test assoc full

 ( 1)  assoc = 0.0
 ( 2)  full = 0.0

       F(  2,   220) =  145.87
            Prob > F =    0.0000
```

```
. test female

 ( 1)   female = 0.0

        F(  1,    220) =    4.88
             Prob > F =    0.0282

. test femassoc femfull

 ( 1)   femassoc = 0.0
 ( 2)   femfull = 0.0

        F(  2,    220) =    1.69
             Prob > F =    0.1876
```

regress followed by the appropriate **test** commands obtains exactly the same R^2 and F test results that we found earlier using **anova**. Predicted values from this regression equal the mean salaries.

```
. predict predpay1
(option xb assumed; fitted values)

. label variable predpay1 "OLS predicted salary"

. table gender rank, contents(mean predpay1)

----------------------------------------
Gender     |
(dummy     |        Academic rank
variable)  |   Assist      Assoc       Full
-----------+----------------------------
      Male |    29280   38622.22    52084.9
    Female | 28711.14   38019.05      47190
----------------------------------------
```

Predicted values (means), R^2, and F tests would also be the same regardless of which categories we chose to omit from the regression. Our "omitted categories," males and assistant professors, are not really absent. Their information is implied by the included categories: if a faculty member is not female, he must be male, and so forth.

To perform a robust analysis of variance, apply **rreg** to this model:

```
. rreg pay assoc full female femassoc femfull, nolog

Robust regression estimates                           Number of obs =       226
                                                      F(  5,    220) =   138.25
                                                      Prob > F      =    0.0000

------------------------------------------------------------------------------
         pay |      Coef.   Std. Err.       t     P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       assoc |  -315.6463   458.1588     -0.69   0.492     -1218.588    587.2956
        full |   9765.296   660.4048     14.79   0.000      8463.767    11066.83
      female |  -749.4949   385.5778     -1.94   0.053     -1509.394    10.40395
    femassoc |   197.7833   458.1588      0.43   0.666     -705.1587    1100.725
     femfull |   -913.348   660.4048     -1.38   0.168     -2214.878    388.1815
       _cons |   38331.87   385.5778     99.41   0.000      37571.97    39091.77
------------------------------------------------------------------------------
```

```
. test assoc full

 ( 1)  assoc = 0.0
 ( 2)  full = 0.0

       F(  2,   220) =  182.67
            Prob > F = .  0.0000

. test female

 ( 1)  female = 0.0

       F(  1,   220) =    3.78
            Prob > F =    0.0532

. test femassoc femfull

 ( 1)  femassoc = 0.0
 ( 2)  femfull = 0.0

       F(  2,   220) =    1.16
            Prob > F =    0.3144
```

rreg downweights several outliers, mainly highly-paid male full professors. To see the robust means, again use predicted values:

```
. predict predpay2
(option xb assumed; fitted values)

. label variable predpay2 "Robust predicted salary"

. table gender rank, contents(mean predpay2)

----------------------------------------
Gender     |
(dummy     |        Academic rank
variable)  |   Assist      Assoc       Full
-----------+----------------------------
      Male | 28916.15   38567.93   49760.01
    Female | 28848.29   37464.51   46434.32
----------------------------------------
```

The male–female salary gap among assistant and full professors appears smaller if we use robust means. It does not entirely vanish, however, and the gender gap among associate professors slightly widens.

With effect coding and suitable interaction terms, **regress** can duplicate ANOVA exactly. **rreg** can do parallel analyses, testing for differences among robust means instead of ordinary means (as **regress** and **anova** do). Used in similar fashion, **qreg** opens the third possibility of testing for differences among medians. For comparison, here is a quantile regression version of the faculty pay analysis:

```
. qreg pay assoc full female femassoc femfull, nolog

Median regression                                    Number of obs =       226
  Raw sum of deviations   1738010 (about 37360)
  Min sum of deviations    798870                    Pseudo R2     =    0.5404

------------------------------------------------------------------------------
         pay |      Coef.   Std. Err.       t     P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       assoc |       -760   440.1693     -1.73   0.086    -1627.488    107.4881
        full |      10335   615.7735     16.78   0.000      9121.43    11548.57
      female |  -623.3333   365.1262     -1.71   0.089    -1342.926     96.2594
    femassoc |  -156.6667   440.1693     -0.36   0.722    -1024.155    710.8214
     femfull |  -691.6667   615.7735     -1.12   0.263    -1905.236    521.9031
       _cons |      38300   365.1262    104.90   0.000     37580.41    39019.59
------------------------------------------------------------------------------

. test assoc full

 ( 1)  assoc = 0.0
 ( 2)  full = 0.0

       F(  2,   220) =  208.94
            Prob > F =    0.0000

. test female

 ( 1)  female = 0.0

       F(  1,   220) =    2.91
            Prob > F =    0.0892

. test femassoc femfull

 ( 1)  femassoc = 0.0
 ( 2)  femfull = 0.0

       F(  2,   220) =    1.60
            Prob > F =    0.2039

. predict predpay3
(option xb assumed; fitted values)

. label variable predpay3 "Median predicted salary"

. table gender rank, contents(mean predpay3)

----------------------------------
Gender    |
(dummy    |     Academic rank
variable) | Assist    Assoc     Full
----------+-----------------------
     Male |  28500    38320    49950
   Female |  28950    36760    47320
----------------------------------
```

Predicted values from this quantile regression closely resemble the median salaries in each subgroup, as we can verify directly:

```
. table gender rank, contents(median pay)

----------------------------------
Gender     |
(dummy     |      Academic rank
variable)  | Assist    Assoc     Full
-----------+----------------------
      Male |  28500    38320    49950
    Female |  28950    36590    46530
----------------------------------
```

qreg thus allows us to fit models analogous to *N*-way ANOVA or ANCOVA, but involving .5 quantiles or approximate medians instead of the usual means. 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.

Further rreg and qreg Applications

Diagnostic statistics and plots (Chapter 7) and nonlinear transformations (Chapter 8) extend the usefulness of robust procedures as they do in ordinary regression. With transformed variables, rreg or qreg fit curvilinear regression models. rreg can also robustly perform simpler types of analysis. To obtain a 90% confidence interval for the mean of a single variable, *y*, we could type either the usual confidence-interval command ci :

```
. ci y, level(90)
```

Or, we could get exactly the same mean and interval through a regression with no *x* variables:

```
. regress y, level(90)
```

Similarly, we can obtain robust mean with 90% confidence interval by typing

```
. rreg y, level(90)
```

qreg could be used in the same way, but keep in mind the previous section's note about how a .5 quantile found by qreg might differ from a sample median. 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) with regression, for example, by regressing the measurement variable on a dummy variable (here called *group*) representing the two categories:

```
. regress y group
```

A robust version of this test results from typing the following command:

```
. rreg y group
```

qreg performs median regression by default, but it is actually a more general tool. It can fit linear models for any quantile of *y*, not just the median (.5 quantile). For example, commands such as the following analyze how the first quartile (.25 quantile) of *y* changes with *x*.

```
. qreg y x, quant(.25)
```

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.

Robust Estimates of Variance — 1

Both **rreg** and **qreg** tend to perform better than OLS (**regress** or **anova**) in the presence of outlier-prone, nonnormal errors. All of these procedures share the common assumption that errors follow independent and identical distributions, however. If the distributions of errors vary across *x* values or observations, then the standard errors calculated by **anova** , **regress** , **rreg** , or **qreg** probably will understate the true sample-to-sample variation, and yield unrealistically narrow confidence intervals.

regress and some other model fitting commands (although not **rreg** or **qreg**) 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. The artificial dataset (*robust2.dta*) provides a first example.

```
Contains data from C:\data\robust2.dta
  obs:           500                          Robust regression examples 2
                                                (artificial data)
 vars:            12                          17 Jul 2005 09:03
 size:        24,500 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
x               float  %9.0g                  Standard normal x
e5              float  %9.0g                  Standard normal errors
y5              float  %9.0g                  y5 = 10 + 2*x + e5 (normal
                                                i.i.d. errors)
e6              float  %9.0g                  Contaminated normal errors:
                                                95% N(0,1), 5%(N(0,10)
y6              float  %9.0g                  y6 = 10 + 2*x + e6
                                                (Contaminated normal errors)
e7              float  %9.0g                  Centered chi-square(1) errors
y7              float  %9.0g                  y7 = 10 + 2*x + e7 (skewed
                                                errors)
e8              float  %9.0g                  Normal errors, variance
                                                increases with x
y8              float  %9.0g                  y8 = 10 + 2*x + e8
                                                (heteroskedasticity)
group           byte   %9.0g
e9              float  %9.0g                  Normal errors, variance
                                                increases with x, mean &
                                                variance increase with cluster
y9              float  %9.0g                  y9 = 10 + 2*x + e9
                                                (heteroskedasticity &
                                                correlated errors)
-------------------------------------------------------------------------------
Sorted by:
```

When we regress *y8* on *x*, we obtain a significant positive slope. A scatterplot shows strong heteroskedasticity, however (Figure 9.5). Variation around the regression line increases with *x*. Because errors do not appear to be identically distributed at all values of *x*, the standard errors, confidence intervals, and tests printed by **`regress`** are untrustworthy. **`rreg`** or **`qreg`** would face the same problem.

```
. regress y8 x

      Source |       SS       df       MS                  Number of obs =     500
-------------+------------------------------               F(  1,   498) =  133.96
       Model |  1607.35658     1  1607.35658               Prob > F      =  0.0000
    Residual |  5975.19162   498  11.9983767               R-squared     =  0.2120
-------------+------------------------------               Adj R-squared =  0.2104
       Total |   7582.5482   499  15.1954874               Root MSE      =  3.4639

------------------------------------------------------------------------------
          y8 |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   1.819032   .1571612     11.57    0.000       1.510251    2.127813
       _cons |   10.06642    .154919     64.98    0.000       9.762047     10.3708
------------------------------------------------------------------------------
```

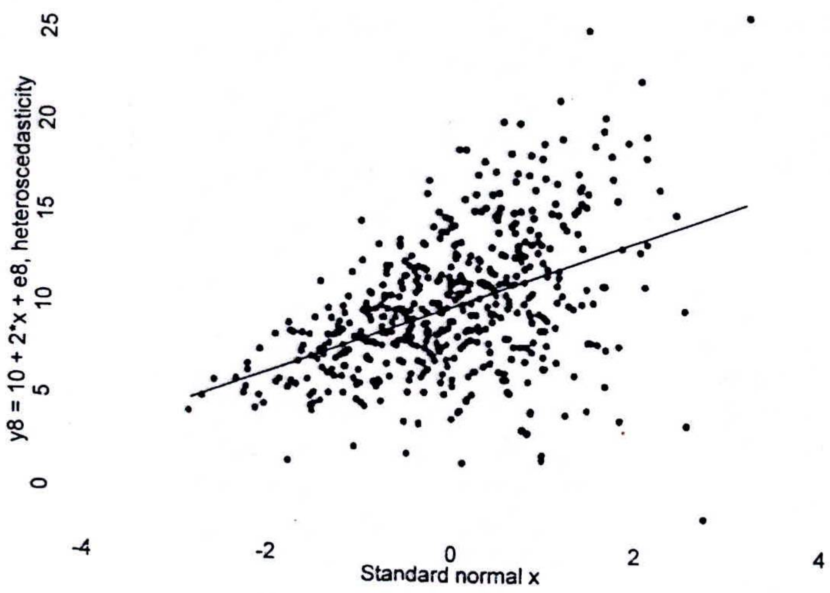


Figure 9.5

More credible standard errors and confidence intervals for this OLS regression can be obtained by using the **`robust`** option:

```
. regress y8 x, robust

Regression with robust standard errors                 Number of obs =      500
                                                       F(  1,    498) =    83.80
                                                       Prob > F      =   0.0000
                                                       R-squared     =   0.2120
                                                       Root MSE      =   3.4639

------------------------------------------------------------------------------
             |               Robust
          y8 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |   1.819032   .1987122     9.15   0.000     1.428614    2.209449
       _cons |   10.06642   .1561846    64.45   0.000     9.759561    10.37328
------------------------------------------------------------------------------
```

Although the fitted model remains unchanged, the robust standard error for the slope is 27% larger (.199 vs. .157) than its nonrobust counterpart. With the **`robust`** option, the regression output does not show the usual ANOVA sums of squares because these no longer have their customary interpretation.

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 likelihood (across repeated sampling) of containing the true value of β. Rather, the robust confidence intervals have a certain likelihood (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.

Robust Estimates of Variance — 2

Another robust-variance option, **`cluster`**, allows us to relax the independent-errors assumption in a limited way, when errors are correlated within subgroups or clusters of the data. The data in *attract.dta* describe an undergraduate social experiment that can be used for illustration. In this experiment, 51 college students were asked to individually rate the attractiveness, on a scale from 1 to 10, of photographs of unknown men and women. The rating exercise was repeated by each participant, given the same photos shuffled in random order, on four occasions during evening social events. 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 participant's (rater's) own gender, and *bac* his or her blood alcohol content at the time, measured by Breathalyzer.

```
Contains data from C:\data\attract.dta
  obs:           204                          Perceived attractiveness and
                                                drinking (D. C. Hamilton 2003)
 vars:             8                          18 Jul 2005 17:27
 size:         5,508 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
id              byte   %9.0g                  Participant number
gender          byte   %9.0g       sex        Participant gender (female)
bac             float  %9.0g                  Blood alchohol content
genbac          float  %9.0g                  gender*bac interaction
relstat         byte   %9.0g       rel        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
```

Although the data contain 204 observations, these represent only 51 individual participants. It seems reasonable to assume that disturbances (unmeasured influences on the ratings) were correlated across the repetitions by each individual. Viewing each participant's four rating sessions as a cluster should yield more realistic standard error estimates. Adding the option **`cluster(`*`id`*`)`** to a regression command, as seen below, obtains robust standard errors across clusters defined by *id* (individual participant).

```
. regress ratefem bac gender genbac, cluster(id)

Regression with robust standard errors                 Number of obs =     204
                                                       F(  3,    50) =    7.75
                                                       Prob > F      =  0.0002
                                                       R-squared     =  0.1264
Number of clusters (id) = 51                           Root MSE      =  1.1219

------------------------------------------------------------------------------
             |               Robust
     ratefem |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         bac |   2.896741   .8543378     3.39   0.001     1.180753    4.612729
      gender |  -.7299683   .3383096    -2.16   0.036    -1.409504   -.0504741
      genbac |   .2080538   1.708146     0.12   0.904    -3.222859    3.638967
       _cons |   6.486767    .229689    28.24   0.000     6.025423     6.94811
------------------------------------------------------------------------------
```

Blood alcohol content (*bac*) has a significant positive effect: as *bac* goes up, predicted attractiveness rating of female photos increases as well. Gender (female) has a negative effect: female participants tended to rate female photos as somewhat less attractive (about .73 lower) than male participants did. The interaction of *gender* and *bac* is weak (.21). The intercept- and slope-dummy variable regression model, approximately

$$\text{predicted } \textit{ratefem} = 6.49 + 2.90\textit{bac} - .73\textit{gender} + .21\textit{genbac}$$

can be reduced for male participants (*gender* = 0) to

$$\begin{aligned} \text{predicted } \textit{ratefem} &= 6.49 + 2.90\textit{bac} - (.73 \times 0) + (.21 \times 0 \times \textit{bac}) \\ &= 6.49 + 2.90\textit{bac} \end{aligned}$$

and for female participants (*gender* = 1) to

$$\begin{aligned} \text{predicted } \textit{ratefem} &= 6.49 + 2.90\textit{bac} - (.73 \times 1) + (.21 \times 1 \times \textit{bac}) \\ &= 6.49 + 2.90\textit{bac} - .73 + .21\textit{bac} \\ &= 5.76 + 3.11\textit{bac} \end{aligned}$$

The slight difference between the effects of alcohol on males (2.90) and females (3.11) equals the interaction coefficient, .21.

Attractiveness ratings for photographs of males were likewise positively affected by blood alcohol content. Gender has a stronger effect on the ratings of male photos: female participants tended to give male photos much higher ratings than male participants did. For male-photo ratings, the *gender* × *bac* interaction is substantial (–4.36), although it falls short of the .05 significance level.

```
. regress ratemal bac gender genbac, cluster(id)

Regression with robust standard errors                 Number of obs =      201
                                                       F(  3,     50) =    10.96
                                                       Prob > F      =   0.0000
                                                       R-squared     =   0.3516
Number of clusters (id) = 51                           Root MSE      =   1.3931

------------------------------------------------------------------------------
             |               Robust
   ratemale  |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        bac  |   4.246042   2.261792     1.88   0.066    -.2969004    8.788985
     gender  |   2.443216   .4529047     5.39   0.000      1.53353    3.352902
     genbac  |  -4.364301   3.573689    -1.22   0.228    -11.54227    2.813663
      _cons  |   3.628043   .2504253    14.49   0.000     3.125049    4.131037
------------------------------------------------------------------------------
```

The regression equation for ratings of male photos by male participants is approximately

$$\begin{aligned} \text{predicted } \textit{ratemale} &= 3.63 + 4.25\textit{bac} + (2.44 \times 0) - (4.36 \times 0 \times \textit{bac}) \\ &= 3.63 + 4.25\textit{bac} \end{aligned}$$

and for rating of male photos by female participants,

$$\begin{aligned} \text{predicted } \textit{ratemale} &= 3.63 + 4.25\textit{bac} + (2.44 \times 1) - (4.36 \times 1 \times \textit{bac}) \\ &= 6.07 - 0.11\textit{bac} \end{aligned}$$

The difference between the substantial alcohol effect on male participants (4.25) and the near-zero alcohol effect on females (–0.11) equals the interaction coefficient, –4.36. In this sample, males' ratings of male photos increase steeply, and females' ratings of male photos remain virtually steady, as the rater's *bac* increases.

Figure 9.6 visualizes these results in a graph. We see positive *rating–bac* relationships across all subplots except for females rating males. The graphs also show other gender differences, including higher *bac* values among male participants.

to female photos. *gender* records the participant's (rater's) own gender, and *bac* his or her blood alcohol content at the time, measured by Breathalyzer.

```
Contains data from C:\data\attract.dta
  obs:           204                         Perceived attractiveness and
                                               drinking (D. C. Hamilton 2003)
 vars:             8                         18 Jul 2005 17:27
 size:         5,508 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
id              byte   %9.0g                  Participant number
gender          byte   %9.0g       sex        Participant gender (female)
bac             float  %9.0g                  Blood alchohol content
genbac          float  %9.0g                  gender*bac interaction
relstat         byte   %9.0g       rel        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
```

Although the data contain 204 observations, these represent only 51 individual participants. It seems reasonable to assume that disturbances (unmeasured influences on the ratings) were correlated across the repetitions by each individual. Viewing each participant's four rating sessions as a cluster should yield more realistic standard error estimates. Adding the option **cluster(*id*)** to a regression command, as seen below, obtains robust standard errors across clusters defined by *id* (individual participant).

```
. regress ratefem bac gender genbac, cluster(id)

Regression with robust standard errors                 Number of obs =     204
                                                       F(  3,    50) =    7.75
                                                       Prob > F      =  0.0002
                                                       R-squared     =  0.1264
Number of clusters (id) = 51                           Root MSE      =  1.1219

------------------------------------------------------------------------------
             |               Robust
     ratefem |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         bac |   2.896741   .8543378     3.39   0.001     1.180753    4.612729
      gender |  -.7299888   .3383096    -2.16   0.036    -1.409504   -.0504741
      genbac |   .2080538   1.708146     0.12   0.904    -3.222859    3.638967
       _cons |   6.486767    .229689    28.24   0.000     6.025423     6.94811
------------------------------------------------------------------------------
```

Blood alcohol content (*bac*) has a significant positive effect: as *bac* goes up, predicted attractiveness rating of female photos increases as well. Gender (female) has a negative effect: female participants tended to rate female photos as somewhat less attractive (about .73 lower) than male participants did. The interaction of *gender* and *bac* is weak (.21). The intercept- and slope-dummy variable regression model, approximately

$$\text{predicted } ratefem = 6.49 + 2.90bac - .73gender + .21genbac$$

can be reduced for male participants ($gender = 0$) to

$$\begin{aligned}\text{predicted } ratefem &= 6.49 + 2.90bac - (.73 \times 0) + (.21 \times 0 \times bac)\\ &= 6.49 + 2.90bac\end{aligned}$$

and for female participants ($gender = 1$) to

$$\begin{aligned}\text{predicted } ratefem &= 6.49 + 2.90bac - (.73 \times 1) + (.21 \times 1 \times bac)\\ &= 6.49 + 2.90bac - .73 + .21bac\\ &= 5.76 + 3.11bac\end{aligned}$$

The slight difference between the effects of alcohol on males (2.90) and females (3.11) equals the interaction coefficient, .21.

Attractiveness ratings for photographs of males were likewise positively affected by blood alcohol content. Gender has a stronger effect on the ratings of male photos: female participants tended to give male photos much higher ratings than male participants did. For male-photo ratings, the *gender* × *bac* interaction is substantial (–4.36), although it falls short of the .05 significance level.

```
. regress ratemal bac gender genbac, cluster(id)

Regression with robust standard errors                 Number of obs =      201
                                                       F(  3,    50) =    10.96
                                                       Prob > F      =   0.0000
Number of clusters (id) = 51                           R-squared     =   0.3516
                                                       Root MSE      =   1.3931

------------------------------------------------------------------------------
             |               Robust
    ratemale |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         bac |   4.246042   2.261792     1.88   0.066    -.2969004    8.788985
      gender |   2.443216   .4529047     5.39   0.000      1.53353    3.352902
      genbac |  -4.364301   3.573689    -1.22   0.228    -11.54227    2.813663
       _cons |   3.628043   .2504253    14.49   0.000     3.125049    4.131037
------------------------------------------------------------------------------
```

The regression equation for ratings of male photos by male participants is approximately

$$\begin{aligned}\text{predicted } ratemale &= 3.63 + 4.25bac + (2.44 \times 0) - (4.36 \times 0 \times bac)\\ &= 3.63 + 4.25bac\end{aligned}$$

and for rating of male photos by female participants,

$$\begin{aligned}\text{predicted } ratemale &= 3.63 + 4.25bac + (2.44 \times 1) - (4.36 \times 1 \times bac)\\ &= 6.07 - 0.11bac\end{aligned}$$

The difference between the substantial alcohol effect on male participants (4.25) and the near-zero alcohol effect on females (–0.11) equals the interaction coefficient, –4.36. In this sample, males' ratings of male photos increase steeply, and females' ratings of male photos remain virtually steady, as the rater's *bac* increases.

Figure 9.6 visualizes these results in a graph. We see positive *rating–bac* relationships across all subplots except for females rating males. The graphs also show other gender differences, including higher *bac* values among male participants.

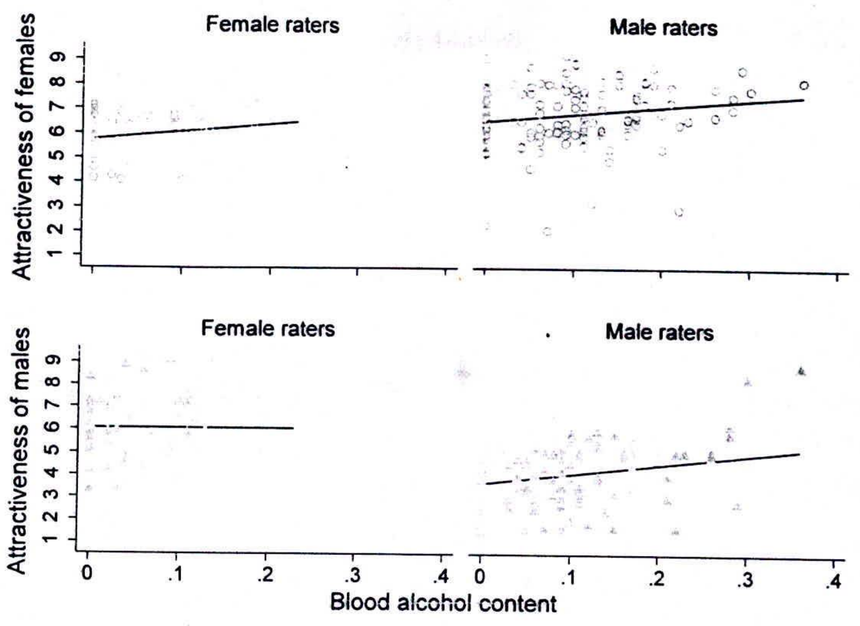


Figure 9.6

OLS regression with robust standard errors, estimated by **`regress`** with the **`robust`** option, should not be confused with the robust regression estimated by **`rreg`**. Despite similar-sounding names, the two procedures are unrelated, and solve different problems.

10

Logistic Regression

The regression and ANOVA methods described in Chapters 5 through 9 require measured dependent or *y* variables. Stata also offers a full range of techniques for modeling categorical, ordinal, and censored dependent variables. A list of some relevant commands follows. For more details on any of these, type **`help`** *command*.

`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.
`cnreg`	Censored-normal regression, assuming that *y* follows a Gaussian distribution but is censored at a point that might vary from observation to observation.
`constraint`	Defines, lists, and drops linear constraints.
`dprobit`	Probit regression giving changes in probabilities instead of coefficients.
`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.
`logistic`	Logistic regression, giving odds ratios.
`logit`	Logistic regression — similar to **`logistic`**, but giving coefficients instead of odds ratios.
`mlogit`	Multinomial logistic regression, with polytomous *y* variable.
`nlogit`	Nested logit estimation.
`ologit`	Logistic regression with ordinal *y* variable.
`oprobit`	Probit regression with ordinal *y* variable.
`probit`	Probit regression, with 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.
`svy: logit`	Logistic regression with complex survey data. Survey (**`svy`**) versions of many other categorical-variables modeling commands also exist.
`tobit`	Tobit regression, assuming *y* follows a Gaussian distribution but is censored at a known, fixed point (see **`cnreg`** for a more general version).
`xtcloglog`	Random-effects and population-averaged cloglog models. Panel (**`xt`**) versions of **`logit`** , **`probit`** , and population-averaged generalized linear models (see **`help xtgee`**) also exist.

After most model-fitting commands, **`predict`** can calculate predicted values or probabilities. **`predict`** also obtains appropriate 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`**).

Examples of several of these commands appear in the next section. Most of the methods for modeling categorical dependent variables can be found under the following menus:

Statistics – Binary outcomes

Statistics – Ordinal outcomes

Statistics – Categorical outcomes

Statistics – Generalized linear models (GLM)

Statistics – Cross-sectional time series

Statistics – Linear regression and related – Censored regression

After the Example Commands section below, the remainder of this chapter concentrates on an important family of methods called logit or logistic regression. We review basic logit methods for dichotomous, ordinal, and polytomous dependent variables.

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 y x1 x2 x3
```

performs essentially the same analysis, but reports effects as logit regression coefficients. The underlying models fit by **`logistic`** and **`logit`** are the same, so subsequent predictions or diagnostic tests will be identical.

```
. lfit
```

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. **`lfit, group(10)`** would perform the test with 10 approximately equal-size groups.

```
. lstat
```

Presents classification statistics and classification table. **`lstat`**, **`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 $y = 1$ based on 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-category 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.

```
. 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.

```
. cnreg y x1 x2 x3, censored(cen)
```

Performs censored-normal regression of measurement variable *y* on three predictors *x1*, *x2*, and *x3*. If an observation's true *y* value is unknown due to left or right censoring, it is replaced for this regression by the nearest *y* value at which censoring occurs. The censoring variable *cen* is a {−1,0,1} indicator of whether each observation's value of *y* has been left censored, not censored, or right censored.

Space Shuttle Data

Our main example for this chapter, *shuttle.dta*, involves 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 last, fatal flight in 1985 (that was 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.

```
Contains data from C:\data\shuttle.dta
  obs:            25                          First 25 space shuttle flights
 vars:             8                          20 Jul 2005 10:40
 size:         1,675 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
flight          byte   %8.0g       flbl       Flight
month           byte   %8.0g                  Month of launch
day             byte   %8.0g                  Day of launch
year            int    %8.0g                  Year of launch
distress        byte   %8.0g       dlbl       Thermal distress incidents
temp            byte   %8.0g                  Joint temperature, degrees F
damage          byte   %9.0g                  Damage severity index (Tufte
                                                1997)
comments        str55  %55s                   Comments (Tufte 1997)
-------------------------------------------------------------------------------
Sorted by:
```

```
. list flight-temp, sepby(year)

     +-------------------------------------------------------------+
     |    flight   month   day   year   date   distress   temp |
     |-------------------------------------------------------------|
  1. |     STS-1       4    12   1981   7772       none     66 |
  2. |     STS-2      11    12   1981   7986     1 or 2     70 |
     |-------------------------------------------------------------|
  3. |     STS-3       3    22   1982   8116       none     69 |
  4. |     STS-4       6    27   1982   8213          .     80 |
  5. |     STS-5      11    11   1982   8350       none     68 |
     |-------------------------------------------------------------|
  6. |     STS-6       4     4   1983   8494     1 or 2     67 |
  7. |     STS-7       6    18   1983   8569       none     72 |
  8. |     STS-8       8    30   1983   8642       none     73 |
  9. |     STS-9      11    28   1983   8732       none     70 |
     |-------------------------------------------------------------|
 10. | STS_41-B        2     3   1984   8799     1 or 2     57 |
 11. | STS_41-C        4     6   1984   8862     3 plus     63 |
 12. | STS_41-D        8    30   1984   9008     3 plus     70 |
 13. | STS_41-G       10     5   1984   9044       none     78 |
 14. | STS_51-A       11     8   1984   9078       none     67 |
     |-------------------------------------------------------------|
 15. | STS_51-C        1    24   1985   9155     3 plus     53 |
 16. | STS_51-D        4    12   1985   9233     3 plus     67 |
 17. | STS_51-B        4    29   1985   9250     3 plus     75 |
 18. | STS_51-G        6    17   1985   9299     3 plus     70 |
 19. | STS_51-F        7    29   1985   9341     1 or 2     81 |
 20. | STS_51-I        8    27   1985   9370     1 or 2     76 |
 21. | STS_51-J       10     3   1985   9407       none     79 |
 22. | STS_61-A       10    30   1985   9434     3 plus     75 |
23. | STS_61-B        11     26    1985    9461      1 or 2       76 |
    |------------------------------------------------------------------|
24. | STS_61-C         1     12    1986    9508      3 plus       58 |
25. | STS_51-L         1     28    1986    9524           .       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 (an arbitrary starting point). *date* is generated from the month, day, and year of launch using the **mdy** (month-day-year to elapsed time; see **help dates**) function:

```
. generate date = mdy(month, day, year)
. label variable date "Date (days since 1/1/60)"
```

Launch 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 the labels, but the **nolabel** option reveals that the underlying numerical codes 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
(2 missing values generated)

. replace any = 1 if distress == 2
(8 real changes made)

. label variable any "Any thermal distress"
```

To see what this accomplished,

```
. tabulate distress any

Thermal     | Any thermal distress
distress    |
incidents   |          0          1 |     Total
------------+----------------------+----------
       none |          9          0 |         9
     1 or 2 |          0          6 |         6
     3 plus |          0          8 |         8
------------+----------------------+----------
      Total |          9         14 |        23
```

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, coef

Iteration 0:   log likelihood = -15.394543
Iteration 1:   log likelihood =  -13.01923
Iteration 2:   log likelihood = -12.991146
Iteration 3:   log likelihood = -12.991096
Logit estimates                                   Number of obs   =         23
                                                  LR chi2(1)      =       4.81
                                                  Prob > chi2     =     0.0283
Log likelihood = -12.991096                       Pseudo R2       =     0.1561

------------------------------------------------------------------------------
         any |      Coef.   Std. Err.       z     P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        date |   .0020907   .0010703     1.95   0.051    -6.93e-06    .0041884
       _cons |  -18.13116   9.517217    -1.91   0.057    -36.78456    .5222396
------------------------------------------------------------------------------
```

The **`logit`** iterative estimation procedure maximizes the logarithm of the likelihood function, shown at the output's top. 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,

$$L = -18.13116 + .0020907\,date \qquad [10.1]$$

where L represents the predicted logit, or log odds, of any distress incidents:

$$L = \ln[P(any = 1) / P(any = 0)] \qquad [10.2]$$

An overall χ^2 test at the upper right evaluates the null hypothesis that all coefficients in the model, except the constant, equal zero,

$$\chi^2 = -2(\ln \mathcal{L}_i - \ln \mathcal{L}_f) \qquad [10.3]$$

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,

$$\chi^2 = -2[-15.394543 - (-12.991096)]$$
$$= 4.81$$

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 Stata's other maximum-likelihood estimation procedures, **`logit`** displays a pseudo R^2 with its output:

$$\text{pseudo } R^2 = 1 - \ln \mathcal{L}_f / \ln \mathcal{L}_i \qquad [10.4]$$

For this example,

$$\text{pseudo } R^2 = 1 - (-12.991096) / (-15.394543)$$
$$= .1561$$

Although they provide a quick way to describe or compare the fit of different models for the same dependent variable, pseudo R^2 statistics lack the straightforward explained-variance interpretation of true R^2 in OLS regression.

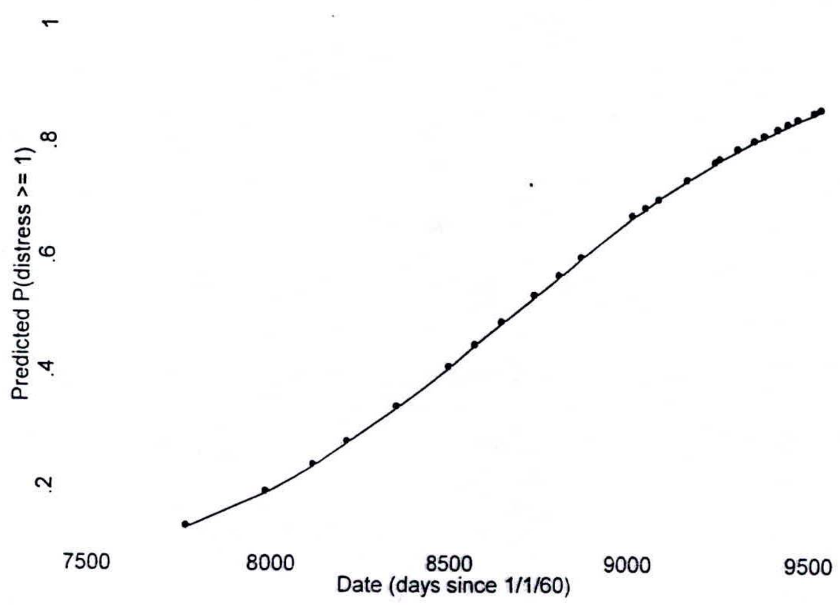
After **`logit`**, the **`predict`** command (with no options) obtains predicted probabilities,

$$Phat = 1 / (1 + e^{-L}) \qquad [10.5]$$

Graphed against *date*, these probabilities follow an S-shaped logistic curve as seen in Figure 10.1.

```
. predict Phat
. label variable Phat "Predicted P(distress >= 1)"
. graph twoway connected Phat date, sort
```

Figure 10.1



The coefficient given by **`logit`** (.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
1.2325359
```

Or, we could simply include an **`or`** (odds ratio) option on the **`logit`** command line. An alternative 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

Logit estimates                                   Number of obs   =         23
                                                  LR chi2(1)      =       4.81
                                                  Prob > chi2     =     0.0283
Log likelihood = -12.991096                       Pseudo R2       =     0.1561

------------------------------------------------------------------------------
         any | Odds Ratio   Std. Err.      z     P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        date |   1.002093   .0010725     1.95   0.051     .9999931    1.004197
------------------------------------------------------------------------------
```

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

```
. lstat

Logistic model for any

              -------- True --------
Classified |         D            ~D  |      Total
-----------+--------------------------+-----------
     +     |        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, **`lstat`** 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.
~D	The event of interest did not occur (that is, $y = 0$) for that observation. In this example, ~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 ≥ .5 given that thermal distress occurred (12 out of 14 or 85.71%).

After **`logistic`** or **`logit`**, the followup command **`predict`** calculates various prediction and diagnostic statistics. Discussion of the 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 jth 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 jth x pattern, h_j
`predict` *`newvar`*`, number`	Assigns numbers to x patterns, $j = 1,2,3 \ldots J$
`predict` *`newvar`*`, resid`	Pearson residual for jth x pattern, r_j
`predict` *`newvar`*`, rstandard`	Standardized Pearson residual

Statistics obtained by the **`dbeta`** , **`dx2`** , **`ddeviance`** , and **`hat`** options do not measure the influence of individual observations, as their counterparts in ordinary regression do. 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) for details. A later section of this chapter shows these statistics in use.

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

Logit estimates                                   Number of obs   =         23
                                                  LR chi2(2)      =       8.09
                                                  Prob > chi2     =     0.0175
Log likelihood = -11.35748                        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 |   .8408319   .0987887    -1.48    0.140     .6678848    1.058561
------------------------------------------------------------------------------
```

The classification table indicates that including temperature as a predictor improved our correct classification rate to 78.26%.

```
. lstat

Logistic model for any

              -------- True --------
Classified |         D            ~D  |      Total
-----------+--------------------------+-----------
     +     |        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                              LR chi2(1)  =       3.28
(Assumption: . nested in full)                     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,

$$\chi^2 = -2(\ln \mathcal{L}_1 - \ln \mathcal{L}_0) \qquad [10.6]$$

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 [10.3]) is a special case of [10.6].

The previous **`lrtest`** example performed this calculation:

$$\chi^2 = -2[-12.991096 - (-11.350748)]$$
$$= 3.28$$

with 1 degree of freedom, yielding $P = .0701$; the effect of *temp* is significant at $\alpha = .10$. Given the small sample and fatal consequences of a Type II error, $\alpha = .10$ seems a more prudent cutoff than the usual $\alpha = .05$.

Conditional Effect Plots

Conditional effect plots help in understanding what a logistic model implies about probabilities. The idea behind such plots is to draw a curve showing how the model's prediction of y changes as a function of one x variable, while holding all other x variables constant at chosen values such as their means, quartiles, or extremes. For example, we could find the predicted probability of any thermal distress incidents as a function of *temp*, holding *date* constant at its 25th percentile. The 25th percentile of *date*, found by **`summarize date, detail`**, is 8569 — that is, June 18, 1983.

```
. quietly logit any date temp
. generate L1 = _b[_cons] + _b[date]*8569 + _b[temp]*temp
. generate Phat1 = 1/(1 + exp(-L1))
. label variable Phat1 "P(distress >= 1 | date = 8569)"
```

L1 is the predicted logit, and *Phat1* equals the corresponding predicted probability that *distress* ≥ 1, calculated according to equation [10.5]. Similar steps find the predicted probability of any distress with *date* fixed at its 75th percentile (9341, or July 29, 1985):

```
. generate L2 = _b[_cons] + _b[date]*9341 + _b[temp]*temp
. generate Phat2 = 1/(1 + exp(-L2))
. label variable Phat2 "P(distress >= 1 | date = 9341)"
```

We can now graph the relationship between *temp* and the probability of any distress, for the two levels of *date*, as shown in Figure 10.2. Using median splines with many vertical bands (**graph twoway mspline, bands(50)**) produces smooth curves in this figure, approximating the smooth logistic functions.

```
. graph twoway mspline Phat1 temp, bands(50)
      || mspline Phat2 temp, bands(50)
      || , ytitle("Probability of thermal distress")
     ylabel(0(.2)1, grid) xlabel(, grid)
        legend(label(1 "June 1983") label(2 "July 1985")
        rows(2) position(7) ring(0))
```

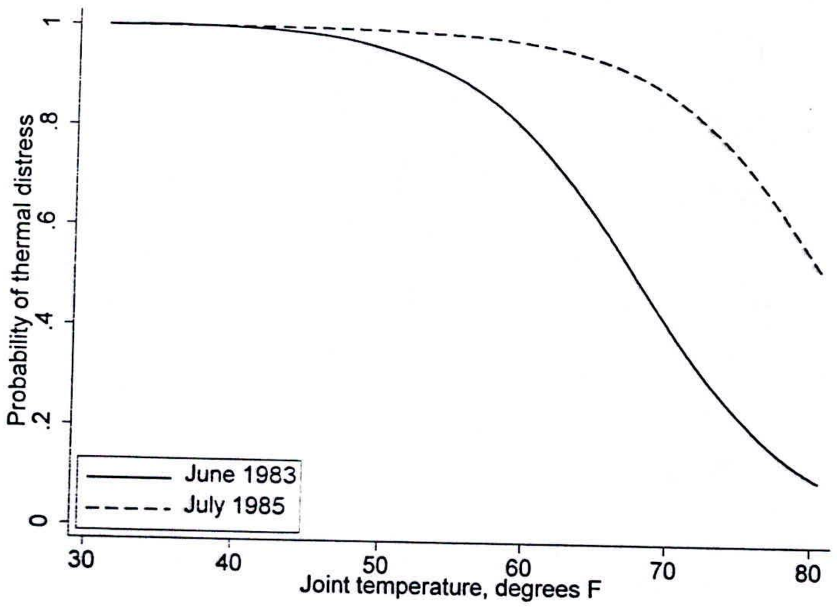


Figure 10.2

Among earlier flights (*date* = 8569, left curve), the probability of thermal distress goes from very low, at around 80° F, to near 1, below 50° F. Among later flights (*date* = 9341, right curve), however, the probability of any distress exceeds .5 even in warm weather, and climbs toward 1 on flights below 70° F. Note that *Challenger*'s launch temperature, 31° F, places it at top left in Figure 10.2. This analysis predicts almost certain booster joint damage.

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. With 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, to be sure that model is what we think:

```
. quietly logistic any date temp

. predict Phat3
(option p assumed; Pr(any))

. label variable Phat3 "Predicted probability"

. predict dX2, dx2
(2 missing values generated)

. label variable dX2 "Change in Pearson chi-squared"

. predict dB, dbeta
(2 missing values generated)

. label variable dB "Influence"

. predict dD, ddeviance
(2 missing values generated)

. 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 10.3), type:

```
. graph twoway scatter dX2 Phat3
```

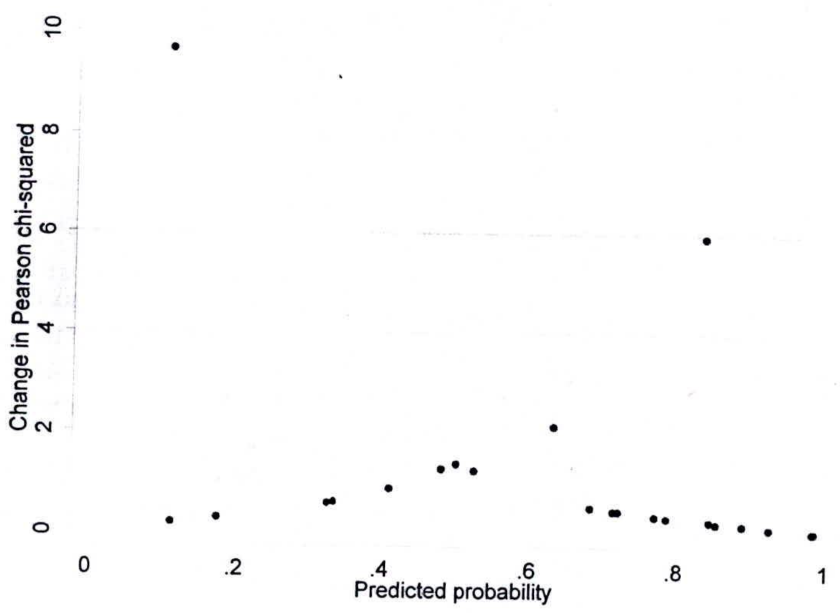
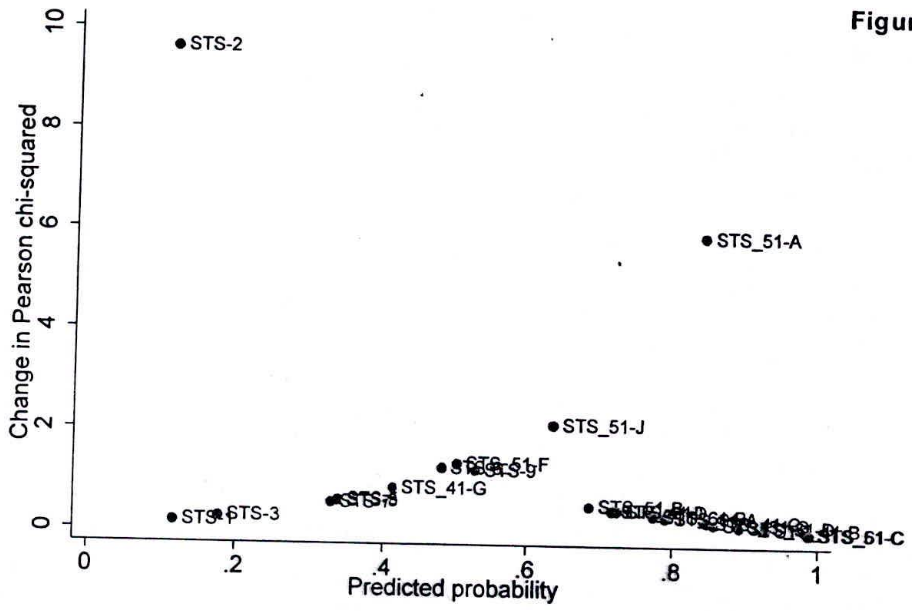


Figure 10.3

Two poorly fit *x* patterns, at upper right and left, stand out. We can identify these two flights (STS-2 and STS 51-A) if we include marker labels in the plot, as seen in Figure 10.4.

```
. graph twoway scatter dX2 Phat3, mlabel(flight) mlabsize(small)
```

Figure 10.4



```
. list flight any date temp dX2 Phat3 if dX2 > 5

     +----------------------------------------------------+
     |   flight   any   date   temp         dX2     Phat3 |
     |----------------------------------------------------|
  2. |    STS-2     1   7986     70    9.630337  .1091805 |
  4. |    STS-4     .   8213     80           .  .0407113 |
 14. | STS_51-A     0   9078     67    5.899742  .8400974 |
 25. | STS_51-L     .   9524     31           .  .9999012 |
     +----------------------------------------------------+
```

Flight STS 51-A experienced no thermal distress, despite a late launch date and cool temperature (see Figure 10.2). The model predicts a .84 probability of distress for this flight. All points along the up-to-right curve in Figure 10.4 have *any* = 0, meaning no thermal distress. 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 considers missing values as "high" numbers, it lists the two missing-values flights, including *Challenger*, among those with *dX2* > 5.)

Similar findings result from plotting *dD* versus predicted probability, as seen in Figure 10.5. Again, flights STS-2 (top left) and STS 51-A (top right) stand out as poorly fit. Figure 10.5 illustrates a variation on the labeled-marker scatterplot. Instead of putting the flight-number labels near the markers, as done earlier in Figure 10.4, we make the markers themselves invisible and place labels where the markers would have been in Figure 10.5.

```
. graph twoway scatter dD Phat3, msymbol(i) mlabposition(0)
      mlabel(flight) mlabsize(small)
```

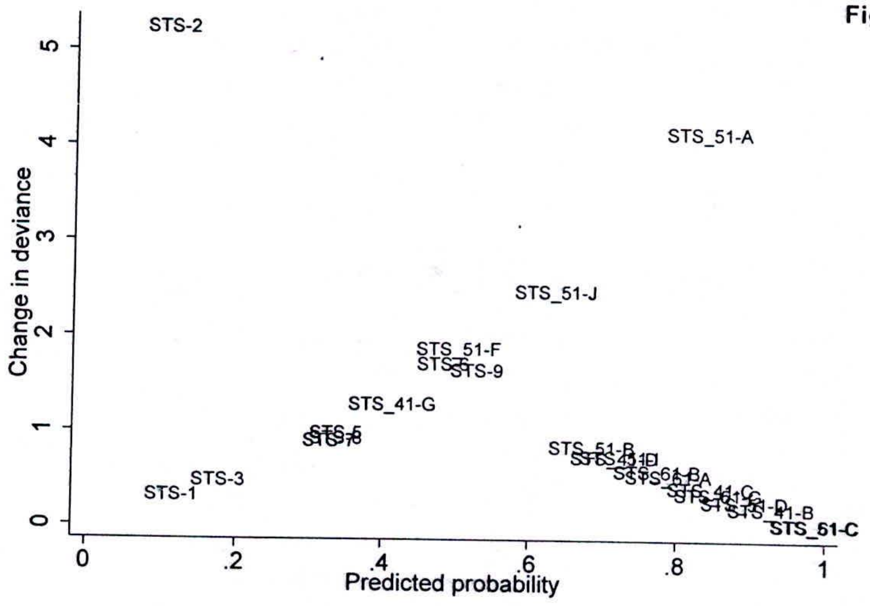


Figure 10.5

dB measures an *x* pattern's influence in logistic regression, as Cook's *D* measures an individual observation's influence in OLS. For a logistic-regression analogue to the OLS diagnostic plot in Figure 7.7, we can make the plotting symbols proportional to influence as done in Figure 10.6. Figure 10.6 reveals that the two worst-fit observations are also the most influential.

```
. graph twoway scatter dD Phat3 [aweight = dB], msymbol(oh)
```

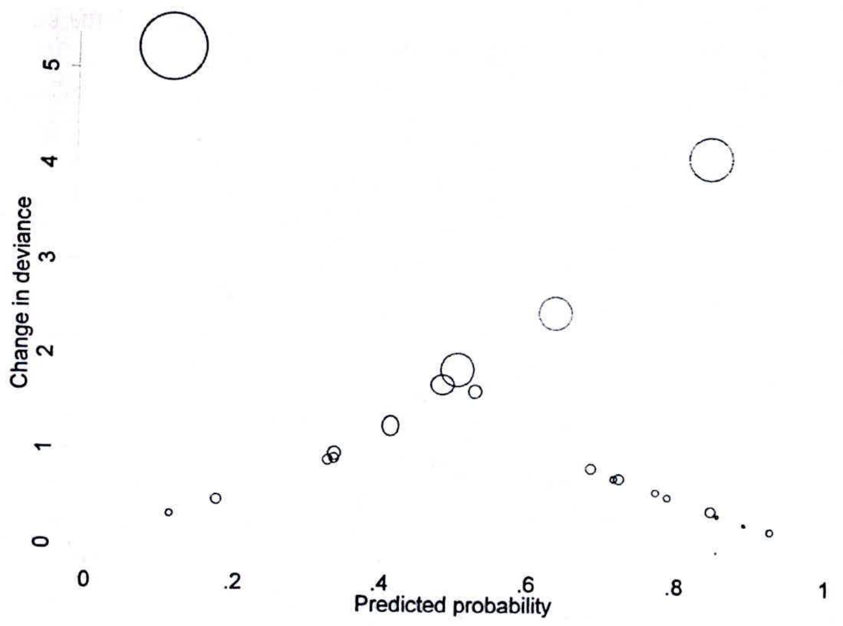


Figure 10.6

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 or to otherwise respecify the model.

Logistic Regression with Ordered-Category *y*

`logit` and **`logistic`** fit only models that have two-category {0,1} *y* variables. We need other methods for models in which *y* takes on more than two categories. For example,

`ologit` Ordered logistic regression, where *y* is an ordinal (ordered-category) variable. The numerical 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 = "undeclared"}.

If *y* is {0,1}, **`logit`** (or **`logistic`**), **`ologit`**, and **`mlogit`** all produce essentially the same estimates.

We earlier simplified the three-category 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 like *distress* that have more than two categories. The numerical codes representing these categories do not matter, so long as higher numerical values mean "more" of whatever is being measured. Recall that *distress* has categories 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 analyses:

```
. ologit distress date temp, nolog

Ordered logit estimates                           Number of obs   =         23
                                                  LR chi2(2)      =      12.32
                                                  Prob > chi2     =     0.0021
Log likelihood =  -18.79706                       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   .0834473    -2.08   0.038     -.336929   -.0098215
-------------+----------------------------------------------------------------
       _cut1 |   16.42813   9.554813          (Ancillary parameters)
       _cut2 |   18.12227   9.722293
------------------------------------------------------------------------------
```

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. Arbitrarily, we can name this model *A*.

```
. estimates store A
```

Next, fit a simpler model without *temp*, store its results as model *B*, and ask for a likelihood-ratio test of whether the fit of reduced model *B* differs significantly from that of the full model, model *A*:

```
. quietly ologit distress date

. estimates store B

. lrtest B A

likelihood-ratio test                                  LR chi2(1)  =      6.12
(Assumption: B nested in A)                            Prob > chi2 =    0.0133
```

The **lrtest** output notes its assumption that model *B* is nested in model *A* — meaning that the parameters estimated in *B* are a subset of those in *A*, 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 *B* 's fit is significantly poorer. Because the presence of *temp* as a predictor in model *A* is the only difference, the likelihood-ratio test thus informs us that *temp*'s contribution is significant. Similar steps find that *date* also has a significant effect.

```
. quietly ologit distress temp

. estimates store C

. lrtest C A

likelihood-ratio test                                  LR chi2(1)  =     10.33
(Assumption: C nested in A)                            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, including more advanced options.

The ordered-logit model estimates a score, *S*, as a linear function of *date* and *temp*:

$$S = .003286 date - .1733752 temp$$

Predicted probabilities depend on the value of *S*, plus a logistically distributed disturbance *u*, relative to the estimated cut points:

$$P(distress=\text{“none”}) = P(S+u \le \_\text{cut1}) = P(S+u \le 16.42813)$$

$$P(distress=\text{“1 or 2”}) = P(\_\text{cut1} < S+u \le \_\text{cut2}) = P(16.42813 < S+u \le 18.12227)$$

$$P(distress=\text{“3 plus”}) = P(\_\text{cut2} < S+u) = P(18.12227 < S+u)$$

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
(option p assumed; predicted probabilities)
```

This creates three new variables:

```
. describe none onetwo threeplus

              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
none            float  %9.0g        .         Pr(distress==0)
onetwo          float  %9.0g                  Pr(distress==1)
threeplus       float  %9.0g                  Pr(distress==2)
```

Predicted probabilities for *Challenger*'s last flight, the 25th in these data, are unsettling:

```
. list flight none onetwo threeplus 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 likelihood 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 more on ordered logistic regression and related techniques. The *Base Reference Manual* explains Stata's implementation.

Multinomial Logistic Regression

When the dependent variable's categories have no natural ordering, we resort to multinomial logistic regression, also called polytomous logistic regression. The **mlogit** command makes this straightforward. If *y* has only two categories, **mlogit** fits the same model as **logistic**. Otherwise, though, an **mlogit** model is more complex. This section presents an extended example interpreting **mlogit** results, using data (*NWarctic.dta*) from a survey of high school students in Alaska's Northwest Arctic borough (Hamilton and Seyfrit 1993).

```
Contains data from C:\data\NWarctic.dta
  obs:           259                          NW Arctic high school students
                                                (Hamilton & Seyfrit 1993)
 vars:             3                          20 Jul 2005 10:40
 size:         2,590 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
life            byte   %8.0g       migrate    Expect to live most of life?
ties            float  %9.0g                  Social ties to community scale
kotz            byte   %8.0g       kotz       Live in Kotzebue or smaller
                                                village?
-------------------------------------------------------------------------------
```

Variable *life* indicates where students say they expect to live most of the rest of their lives: in the same region (Northwest Arctic), elsewhere in Alaska, or outside of Alaska:

```
. tabulate life, plot
```

```
   Expect to |
   live most |
    of life? |      Freq.
-------------+------------+-----------------------------------------------------
        same |         92 |****************************************
    other AK |        120 |****************************************************
    leave AK |         47 |*********************
-------------+------------+-----------------------------------------------------
       Total |        259
```

Kotzebue (population near 3,000) is the Northwest Arctic's regional hub and largest city. More than a third of these students live in Kotzebue. The rest live in smaller villages of 200 to 700 people. The relatively cosmopolitan Kotzebue students less often expect to stay where they are, and lean more towards leaving the state:

```
. tabulate life kotz, chi2

  Expect to |  Live in Kotzebue or
  live most |   smaller village?
   of life? |   village   Kotzebue |     Total
------------+----------------------+----------
       same |        75         17 |        92
   other AK |        80         40 |       120
   leave AK |        11         36 |        47
------------+----------------------+----------
      Total |       166         93 |       259

          Pearson chi2(2) =  46.2992   Pr = 0.000
```

mlogit can replicate this simple analysis (although its likelihood-ratio chi-squared need not exactly equal the Pearson chi-squared found by **tabulate**):

```
. mlogit life kotz, nolog base(1) rrr

Multinomial logistic regression                    Number of obs   =        259
                                                   LR chi2(2)      =      46.23
                                                   Prob > chi2     =     0.0000
Log likelihood = -244.64465                        Pseudo R2       =     0.0863

------------------------------------------------------------------------------
        life |        RRR   Std. Err.       z     P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
other AK     |
        kotz |   2.205882   .7304664     2.39   0.017     1.152687    4.221369
-------------+----------------------------------------------------------------
leave AK     |
        kotz |    14.4385   6.307555     6.11   0.000     6.132946    33.99188
------------------------------------------------------------------------------
(Outcome life==same is the comparison group)
```

base(1) specifies that category 1 of y (*life* = "same") is the base category for comparison. The **rrr** option instructs **mlogit** to show relative risk ratios, which resemble the odds ratios given by **logistic**.

Referring back to the **tabulate** output, we can calculate that among Kotzebue students the odds favoring "leave Alaska" over "stay in the same area" are

$$P(\text{leave AK}) / P(\text{same}) = (36/93) / (17/93) = 2.1176471$$

Among other students the odds favoring "leave Alaska" over "same area" are

$$P(\text{leave AK}) \,/\, P(\text{same}) = (11/166) \,/\, (75/166) = .1466667$$

Thus, the odds favoring "leave Alaska" over "same area" are 14.4385 times higher for Kotzebue students than for others:

$$2.1176471 \,/\, .1466667 = 14.4385$$

This multiplier, a ratio of two odds. equals the relative risk ratio (14.4385) displayed by **`mlogit`**.

In general, the relative risk ratio for category *j* of *y*, and predictor x_k, equals the amount by which predicted odds favoring $y=j$ (compared with $y=$ 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,

$$\text{rrr}_{jk} \times \frac{P(y=j \mid x_k)}{P(y=\text{base} \mid x_k)} = \frac{P(y=j \mid x_k+1)}{P(y=\text{base} \mid x_k+1)}$$

ties is a continuous scale indicating the strength of students' social ties to family and community. We include *ties* as a second predictor:

```
. mlogit life kotz ties, nolog base(1) rrr

Multinomial logistic regression                   Number of obs   =        259
                                                  LR chi2(4)      =      91.96
                                                  Prob > chi2     =     0.0000
Log likelihood = -221.77969                       Pseudo R2       =     0.1717

------------------------------------------------------------------------------
        life |        RRR   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
other AK     |
        kotz |   2.214184    .7724996     2.28   0.023     1.117483    4.387193
        ties |   .4802486    .0799184    -4.41   0.000     .3465911    .6654492
-------------+----------------------------------------------------------------
leave AK     |
        kotz |   14.84604    7.146824     5.60   0.000     5.778907    38.13955
        ties |    .230262     .059095    -5.72   0.000     .1392531      .38075
------------------------------------------------------------------------------
(Outcome life==same is the comparison group)
```

Asymptotic *z* tests here indicate that the four relative risk ratios, describing two *x* variables' effects, all differ significantly from 1.0. If a *y* variable has *J* categories, then **`mlogit`** models the effects of each predictor (*x*) variable with $J-1$ relative risk ratios or coefficients, and hence also employs $J-1$ *z* tests — evaluating two or more separate null hypotheses for each predictor. Likelihood-ratio tests evaluate the overall effect of each predictor. First, store the results from the full model, here given the name *full*:

```
. estimates store full
```

Then fit a simpler model with one of the *x* variables omitted, and perform a likelihood-ratio test. For example, to test the effect of *ties*, we repeat the regression with *ties* omitted:

```
. quietly mlogit life kotz

. estimates store no_ties

. lrtest no_ties full
```

```
likelihood-ratio test                                 LR chi2(2)  =     45.73
(Assumption: no_ties nested in full)                  Prob > chi2 =    0.0000
```

The effect of *ties* is clearly significant. Next, we run a similar test on the effect of *kotz*:

```
. quietly mlogit life ties

. estimates store no_kotz

. lrtest no_kotz full
```

```
likelihood-ratio test                                 LR chi2(2)  =     39.05
(Assumption: no_kotz nested in full)                  Prob > chi2 =    0.0000
```

If our data contained missing values, the three **mlogit** commands just shown might have analyzed three overlapping subsets of observations. The full model would use only observations with nonmissing *life*, *kotz*, and *ties* values; the *kotz*-only model would bring back in any observations missing just their *ties* values; and the *ties*-only model would bring back observations missing just *kotz* values. When this happens, Stata returns an error messages saying "observations differ." In such cases, the likelihood-ratio test would be invalid. Analysts must either screen observations with **if** qualifiers attached to modeling commands, such as

```
. mlogit life kotz ties, nolog base(1) rrr

. estimates store full

. quietly mlogit life kotz if ties < .

. estimates store no_ties

. lrtest no_ties full

. quietly mlogit life ties if kotz < .

. estimates store no_kotz

. lrtest no_kotz full
```

or simply drop all observations having missing values before proceeding:

```
. drop if life >= . | kotz >= . | ties >= .
```

Dataset *NWarctic.dta* has already been screened in this fashion to drop observations with missing values.

Both *kotz* and *ties* significantly predict *life*. What else can we say from this output? To interpret specific effects, recall that *life* = "same" is the base category. The relative risk ratios tell us that:

> Odds that a student expects migration to elsewhere in Alaska rather than staying in the same area are 2.21 times greater (increase about 121%) among Kotzebue students (*kotz*=1), adjusting for social ties to community.
>
> Odds that a student expects to leave Alaska rather than stay in the same area are 14.85 times greater (increase about 1385%) among Kotzebue students (*kotz*=1), adjusting for social ties to community.
>
> Odds that a student expects migration to elsewhere in Alaska rather than staying are multiplied by .48 (decrease about 52%) with each 1-unit (since *ties* is standardized, its units equal standard deviations) increase in social ties, controlling for Kotzebue/village residence.

Odds that a student expects to leave Alaska rather than staying are multiplied by .23 (decrease about 77%) with each 1-unit increase in social ties, controlling for Kotzebue/village residence.

predict can calculate predicted probabilities from **mlogit**. The **outcome(#)** option specifies for which *y* category we want probabilities. For example, to get predicted probabilities that *life* = "leave AK" (category 3),

```
. quietly mlogit life kotz ties

. predict PleaveAK, outcome(3)
(option p assumed; predicted probability)

. label variable PleaveAK "P(life = 3 | kotz, ties)"
```

Tabulating predicted probabilities for each value of the dependent variable shows how the model fits:

```
. table life, contents(mean PleaveAK) row

--------------------------
Expect to  |
live most  |
of life?   | mean(PleaveAK)
-----------+--------------
      same |       .0811267
  other AK |       .1770225
  leave AK |       .3892264
           |
     Total |       .1814672
--------------------------
```

A minority of these students (47/259 = 18%) expect to leave Alaska. The model averages only a .39 probability of leaving Alaska even for those who actually chose this response — reflecting the fact that although our predictors have significant effects, most variation in migration plans remains unexplained.

Conditional effect plots help to visualize what a model implies regarding continuous predictors. We can draw them using estimated coefficients (not risk ratios) to calculate probabilities:

```
. mlogit life kotz ties, nolog base(1)

Multinomial logistic regression                   Number of obs   =        259
                                                  LR chi2(4)      =      91.96
                                                  Prob > chi2     =     0.0000
Log likelihood = -221.77969                       Pseudo R2       =     0.1717

------------------------------------------------------------------------------
        life |      Coef.   Std. Err.       z     P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
other AK     |
        kotz |    .794884    .3488368     2.28    0.023     .1110784     1.47869
        ties |  -.7334513    .1664104    -4.41    0.000     -1.05961    -.407293
       _cons |    .206402    .1728353     1.19    0.232    -.1322902    .5450942
-------------+----------------------------------------------------------------
leave AK     |
        kotz |   2.697733    .4813959     5.60    0.000     1.754215    3.641252
        ties |  -1.468537    .2565991    -5.72    0.000    -1.971462   -.9656124
       _cons |  -2.115025    .3758163    -5.63    0.000    -2.851611   -1.378439
------------------------------------------------------------------------------
(Outcome life==same is the comparison group)
```

The following commands calculate predicted logits, and then the probabilities needed for conditional effect plots. *L2villag* represents the predicted logit of *life* = 2 (other Alaska) for village students. *L3kotz* is the predicted logit of *life* = 3 (leave Alaska) for Kotzebue students, and so forth:

```
. generate L2villag = .206402 +.794884*0 -.7334513*ties
. generate L2kotz = .206402 +.794884*1 -.7334513*ties
. generate L3villag = -2.115025 +2.697733*0 -1.468537*ties
. generate L3kotz = -2.115025 +2.697733*1 -1.468537*ties
```

Like other Stata modeling commands, **mlogit** saves coefficient estimates as macros. For example, [2]_b[*kotz*] refers to the coefficient on *kotz* in the model's second (*life* = 2) equation. Therefore, we could have generated the same predicted logits as follows. *L2v* will be identical to *L2villag* defined earlier, *L3k* the same as *L3kotz*, and so forth:

```
. generate L2v = [2]_b[_cons] +[2]_b[kotz]*0 +[2]_b[ties]*ties
. generate L2k = [2]_b[_cons] +[2]_b[kotz]*1 +[2]_b[ties]*ties
. generate L3v = [3]_b[_cons] +[3]_b[kotz]*0 + [3]_b[ties]*ties
. generate L3k = [3]_b[_cons] +[3]_b[kotz]*1 + [3]_b[ties]*ties
```

From either set of logits, we next calculate the predicted probabilities:

```
. generate P1villag = 1/(1 +exp(L2villag) +exp(L3villag))
. label variable P1villag "same area"
. generate P2villag = exp(L2villag)/(1+exp(L2villag)+exp(L3villag))
. label variable P2villag "other Alaska"
. generate P3villag = exp(L3villag)/(1+exp(L2villag)+exp(L3villag))
. label variable P3villag "leave Alaska"
. generate P1kotz = 1/(1 +exp(L2kotz) +exp(L3kotz))
. label variable P1kotz "same area"
. generate P2kotz = exp(L2kotz)/(1 +exp(L2kotz) +exp(L3kotz))
. label variable P2kotz "other Alaska"
. generate P3kotz = exp(L3kotz)/(1 +exp(L2kotz) +exp(L3kotz))
. label variable P3kotz "leave Alaska"
```

Figures 10.7 and 10.8 show conditional effect plots for village and Kotzebue students separately.

```
. graph twoway mspline P1villag ties, bands(50)
      ||  mspline P2villag ties, bands(50)
      ||  mspline P3villag ties, bands(50)
      ||  , xlabel(-3(1)3) ylabel(0(.2)1) yline(0 1) xline(0)
      legend(order(2 3 1) position(12) ring(0) label(1 "same area")
      label(2 "elsewhere Alaska") label(3 "leave Alaska") cols(1))
      ytitle("Probability")
```

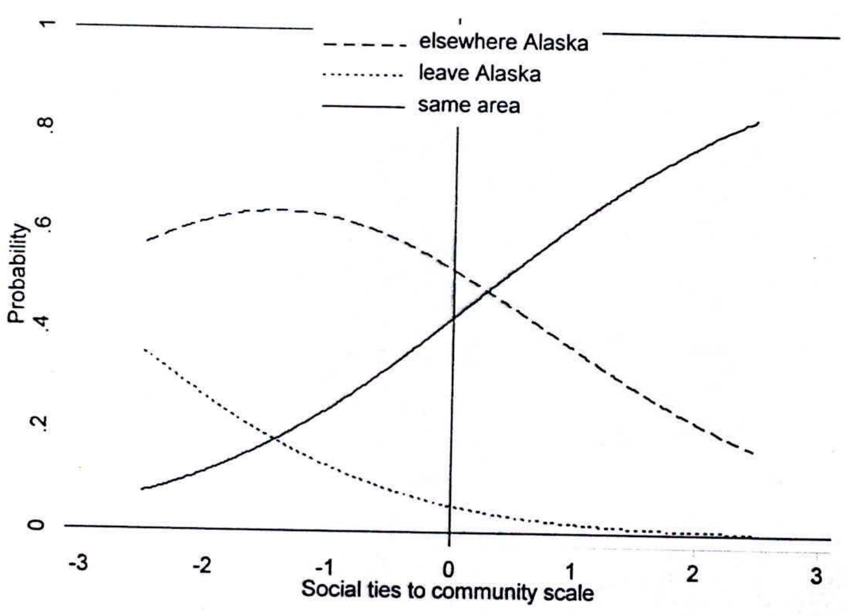


Figure 10.7

```
. graph twoway mspline P1kotz ties, bands(50)
     || mspline P2kotz ties, bands(50)
     || mspline P3kotz ties, bands(50)
     || , xlabel(-3(1)3) ylabel(0(.2)1) yline(0 1) xline(0)
     legend(order(3 2 1) position(12) ring(0) label(1 "same area")
     label(2 "elsewhere Alaska") label(3 "leave Alaska") cols(1))
     ytitle("Probability")
```

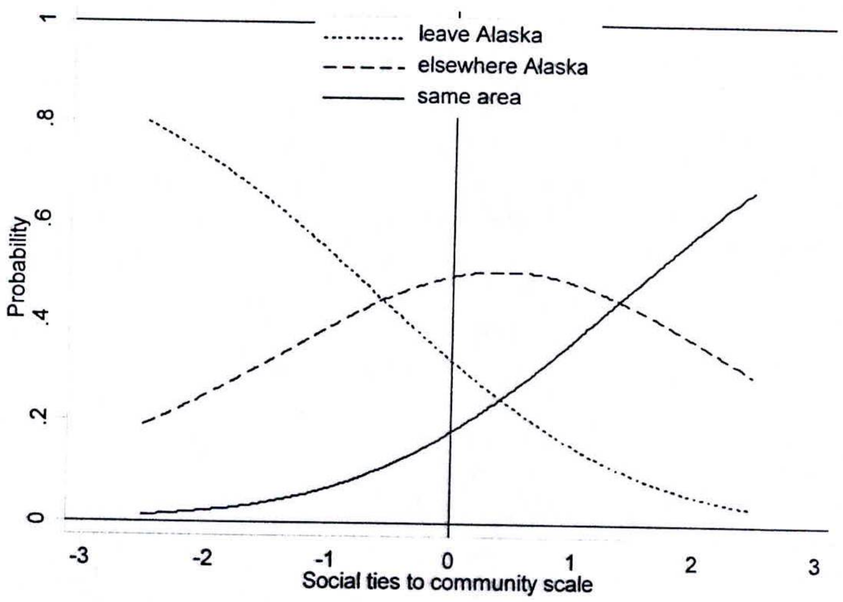


Figure 10.8

The plots indicate that among village students, social ties increase the probability of staying rather than moving elsewhere in Alaska. Relatively few village students expect to leave Alaska. In contrast, among Kotzebue students, *ties* particularly affects the probability of leaving Alaska, rather than simply moving elsewhere in the state. Only if they feel very strong social ties do Kotzebue students tend to favor staying put.

11

Survival and Event-Count Models

This chapter presents methods for analyzing event data..*Survival analysis* encompasses 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 that 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, only 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 Analsysis and Epidemiological Tables Reference Manual* for more information about Stata's capabilities. Type **`help st`** to see an online overview. Selvin (1995) 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, Gould and Gutierrez (2004), a chapter in Rosner (1995), and comprehensive treatments by Hosmer and Lemeshow (1999) 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 has some material on generalized linear models. An extensive and current treatment of generalized linear models is found in Hardin and Hilbe (2001).

Stata menu groups most relevant to this chapter include:

Statistics – Survival analysis

Graphics – Survival analysis graphs

Statistics – Count outcomes

Statistics – Generalized linear models (GLM)

Regarding epidemiological tables, not covered in this chapter, further information can be found by typing **`help epitab`** or exploring the menus for

Statistics – Observational/Epi. analysis.

Example Commands

Most of Stata's survival-analysis (`st*`) commands require that the data have previously been identified as survival-time by issuing an `stset` command (see following). `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.

```
. stdes
```

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 a `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 the `by()` option,

```
. sts graph, by(sex)
```

To adjust, through Cox regression, for the effects of a continuous independent variable such as *age*, use the `adjustfor()` option,

```
. sts graph, by(sex) adjustfor(age)
```

Note: the `by()` and `adjustfor()` options work similarly with the other `sts` commands `sts list`, `sts generate`, and `sts test`.

```
. sts list
```

Lists the estimated Kaplan–Meier survivor (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*–*x3*.

```
. stcox x1 x2 x3, strata(x4) basechazard(hazard) robust
```

Fits a Cox proportional hazard model, stratified by *x4*. Stores the group-specific baseline cumulative hazard function as a new variable named *hazard*. (Baseline survivor function estimates could be obtained through a **basesur(*survive*)** option.) Obtains robust standard error estimates. See Chapter 9 or, for a more complete explanation of robust standard errors, consult the *User's Guide*.

```
. stphplot, by(sex)
```

Plots –ln(–ln(survival)) versus ln(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 **stphtest** (performs test based on Schoenfeld residuals). See **help stcox** 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) 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) lnoffset(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 to each observation. The literature often terms this event of interest a "failure," regardless of its substantive 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 dummy indicating whether the observation failed or was censored. The dataset can also contain any number of other measurement or categorical variables, and 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 and save in Stata format as file *aids1.dta*:

```
. infile case time aids age using aids.raw, clear
(51 observations read)

. 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
case was float now byte
time was float now byte
aids was float now byte
age was float now byte

. save aids1
file c:\data\aids1.dta saved
```

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 data again 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
------------------------------------------------------------------------------
       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, replace
file c:\data\aids1.dta saved
```

stdes 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.

```
. stdes
         failure _d:  aids
   analysis time _t:  time
                 id:  case

                              |-------------- per subject --------------|
Category                total        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

         |               incidence       no. of    |------ Survival time -----|
         | time at risk     rate        subjects        25%       50%       75%
---------+---------------------------------------------------------------------
   total |         3164   .0079014           51          41        81         .
```

If the data happen to include a grouping or categorical variable such as *sex* (0 = male, 1 = female), 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 *aids1.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.

```
Contains data from C:\data\diskdriv.dta
  obs:                 6                          Count-time data on disk drives
 vars:                 3                          21 Jul 2005 09:34
 size:                48 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
hours           int    %8.0g                  Hours of continuous operation
failures        byte   %8.0g                  Number of failures observed
censored        byte   %9.0g                  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

(data are now st)

     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   w   _st   _d     _t   _t0 |
     |------------------------------------------------------|
  1. |  1200          0   5     1    0   1200     0 |
  2. |   200          1   2     1    1    200     0 |
  3. |   400          1   3     1    1    400     0 |
  4. |   600          1   4     1    1    600     0 |
  5. |   800          1   8     1    1    800     0 |
     |------------------------------------------------------|
  6. |  1000          1   3     1    1   1000     0 |
     +------------------------------------------------------+

. stdes

         failure _d:  failures
   analysis time _t:  hours
             weight:  [fweight=w]

                                   |-------------- per subject --------------|
                       unweighted  unweighted              unweighted
Category                    total        mean         min      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 time:  hours
       failure/censor:  failures
               weight:  [fweight=w]

         |               incidence       no. of     |---- Survival time ----|
         | time at risk     rate        subjects        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:

$$S(t) = \prod_{j=t0}^{t} \{(n_j - d_j)/n_j\} \qquad [11.1]$$

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 11.1. Stata draws such graphs automatically with the **sts graph** command. For example,

```
. use aids, clear
(AIDS (Selvin 1995:453))

. sts graph

         failure _d:  aids
   analysis time _t:  time
                 id:  case
```

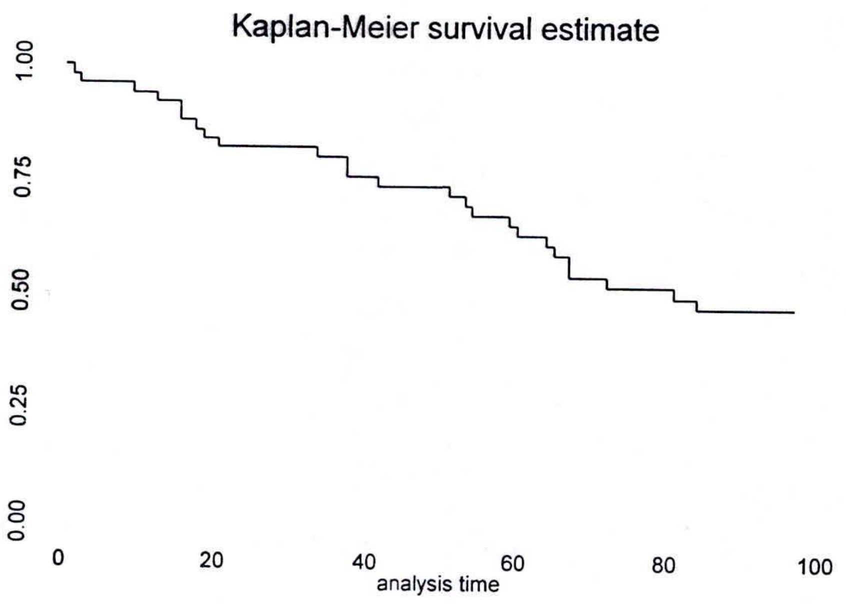


Figure 11.1

For a second example of survivor functions, we turn to data in *smoking1.dta*, adapted from Rosner (1995). The observations are 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:

```
Contains data from C:\data\smoking1.dta
  obs:           234                          Smoking (Rosner 1995:607)
 vars:             8                          21 Jul 2005 09:35
 size:         3,744 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      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

         |               incidence       no. of    |------ Survival time -----|
sex      | time at risk     rate        subjects        25%       50%       75%
---------+---------------------------------------------------------------------
    Male |         8813   .0105526          110          4        15        68
  Female |        10133   .0106582          124          4        15        91
---------+---------------------------------------------------------------------
   total |        18946   .0106091          234          4        15        73
```

Figure 11.2 confirms this similarity, showing 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 beyond a full year.

```
. sts graph, by(sex)

         failure _d:  smoking
   analysis time _t:  days
```

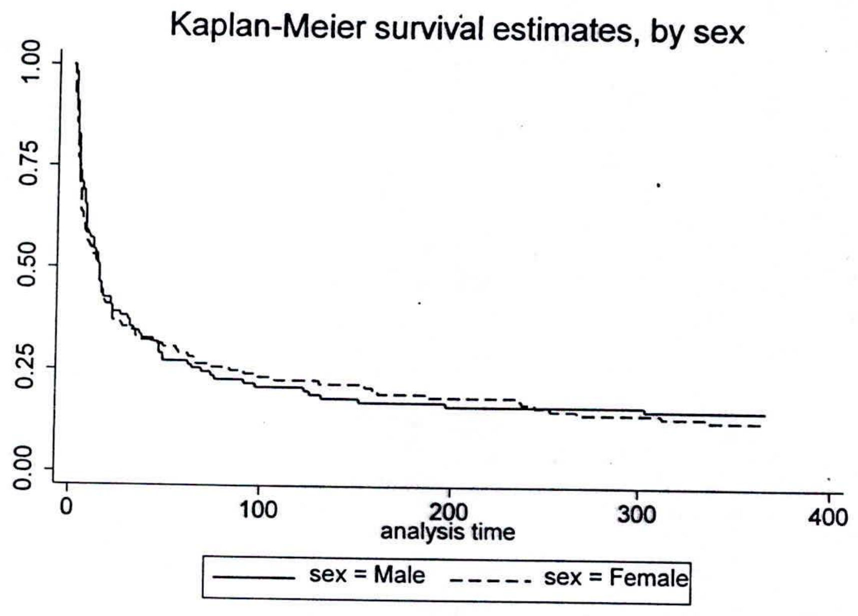


Figure 11.2

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

       |   Events         Events
sex    |  observed       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

$$h(t) = \frac{\text{probability of failing between times } t \text{ and } t + \Delta t}{(\Delta t)\,(\text{probability of failing after time } t)} \qquad [11.2]$$

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,

$$h(t) = h_0(t)\exp(\beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k) \qquad [11.3a]$$

or, equivalently,

$$\ln[h(t)] = \ln[h_0(t)] + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k \qquad [11.3b]$$

"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 [11.3]. Stata's **stcox** procedure ordinarily reports hazard ratios, which are estimates of exp(β). These indicate proportional changes relative to the baseline hazard rate.

Does age affect the onset of AIDS symptoms? Dataset *aids.dta* contains information that helps answer 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, timevariables, and censoring variables are understood automatically with **stset** data.

```
. use aids
(AIDS (Selvin 1995:453))

. 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 ($P = .020$) 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
Time at risk    =         3164
                                                LR chi2(1)      =       5.00
Log likelihood  =   -86.576295                  Prob > chi2     =     0.0254

------------------------------------------------------------------------------
          _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 during this time (*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

              storage  display     value
variable name   type   format      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
```

```
. stdes

         failure _d:  coronary
   analysis time _t:  time

                                   |-------------- per subject --------------|
Category                   total        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
Time at risk    =        90322
                                                   LR chi2(5)       =     13.97
Log likelihood  =   -17.263231                     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
------------------------------------------------------------------------------
```

After estimating the model, **stcox** can also generate 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 recenter 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 100, 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
(35 real changes made)

. replace sbp = sbp - 100
(35 real changes made)

. replace chol = chol - 340
(35 real changes made)

. 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    29.71429    14.28403          4         54
-------------+-----------------------------------------------------
        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 more substantive sense. To create new variables holding the baseline survivor and cumulative hazard function estimates, we repeat the regression with **basesurv()** and **basechaz()** options:

```
. stcox weight sbp chol cigs ab, noshow nolog basesurv(survivor)
      basechaz(hazard)

Cox regression -- no ties

No. of subjects =           35                 Number of obs   =         35
No. of failures =            8
Time at risk    =        90322
                                               LR chi2(5)      =      13.97
Log likelihood  =   -17.263231                 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 recentering three *x* variables had no effect on the hazard ratios, standard errors, and so forth. The command created two new variables, arbitrarily named *survivor* and *hazard*. To graph the baseline survivor function, we plot *survivor* against *time* and connect data points with in a stairstep fashion, as seen in Figure 11.3.

```
. graph twoway line survivor time, connect(stairstep) sort
```

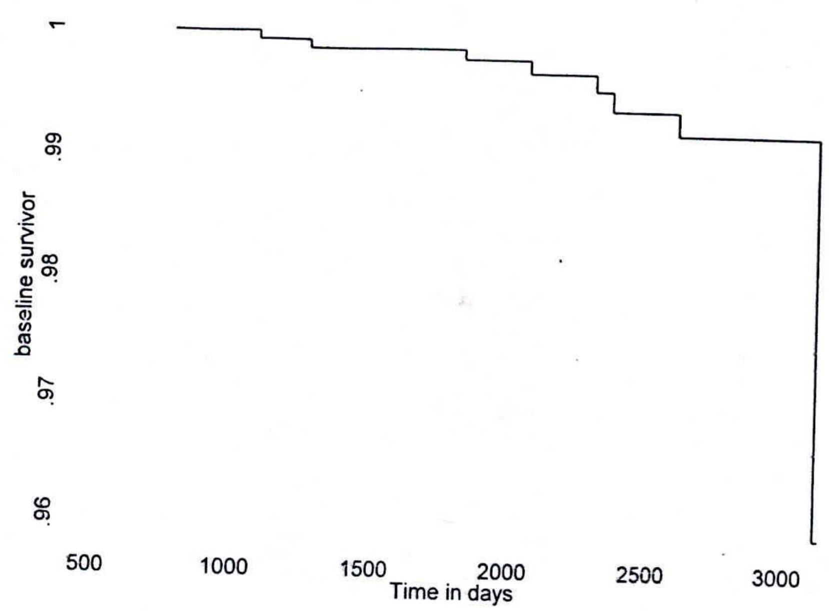


Figure 11.3

The baseline survivor function — which depicts survival probabilities for patients having "0" weight (120 pounds), "0" blood pressure (100), "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 11.4, employs an `sts graph` command with `adjustfor()` option listing the predictor variables:

```
. sts graph, adjustfor(weight sbp chol cigs ab)

         failure _d:  coronary
   analysis time _t:  time
```

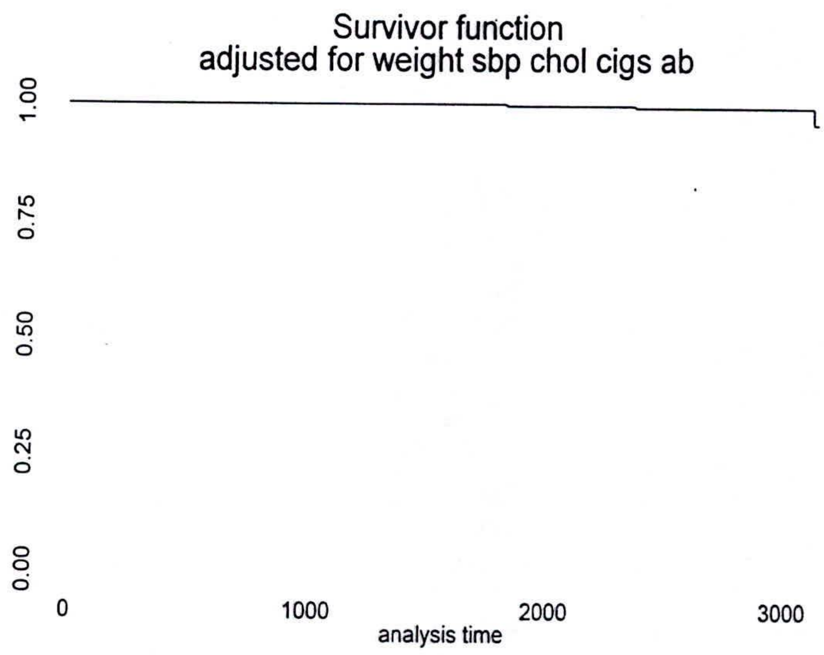


Figure 11.4

Figure 11.4, unlike Figure 11.3, follows the usual survivor-function convention of scaling the vertical axis from 0 to 1. Apart from this difference in scaling, Figures 11.3 and 11.4 depict the same curve.

Figure 11.5 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)
```

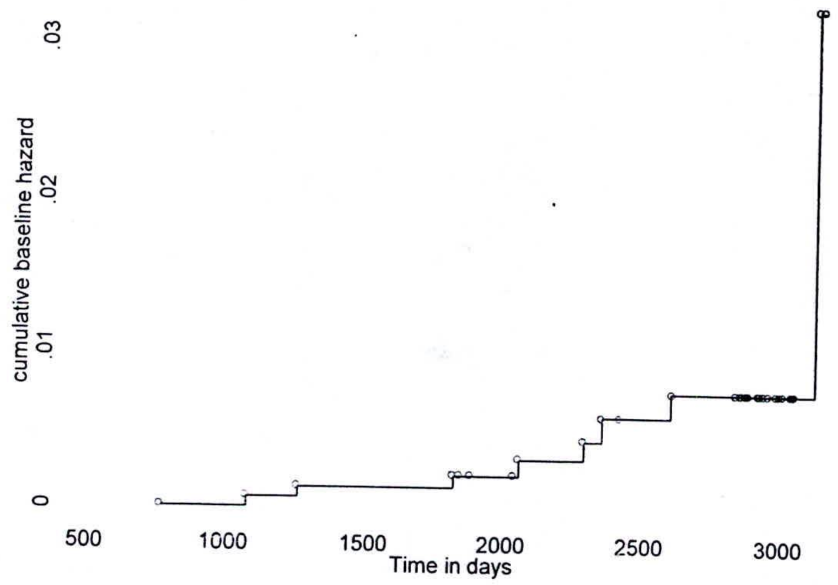


Figure 11.5

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 [11.2] and [11.3]), but define the baseline hazard $h_0(t)$ differently. Two examples appear in this section.

If failures occur randomly, 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 11.6) of $\ln(S(t))$ versus time, after first generating Kaplan–Meier estimates of the survivor function $S(t)$. The *y*-axis labels in Figure 11.6 are given a fixed two-digit, one-decimal display format (%2.1f) and oriented horizontally, to improve their readability.

```
. use aids, clear
(AIDS (Selvin 1995:453))

. sts gen S = S

. generate logS = ln(S)

. graph twoway scatter logS time,
      ylabel(-.8(.1)0, format(%2.1f) angle(horizontal))
```

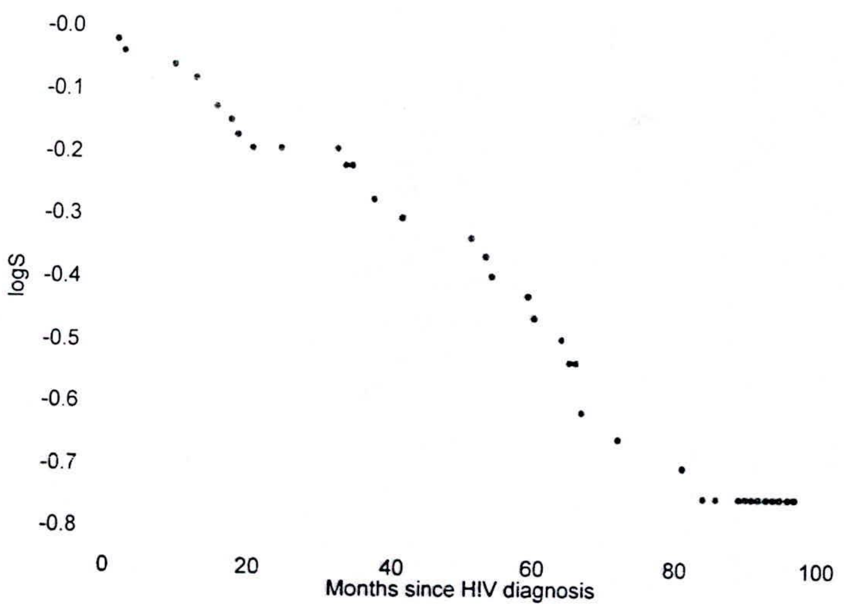


Figure 11.6

The pattern in Figure 11.6 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
Time at risk    =         3164
                                                LR chi2(1)       =      4.34
Log likelihood  =   -59.996976                  Prob > chi2      =    0.0372

------------------------------------------------------------------------------
          _t | Haz. Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |   1.074414   .0349626     2.21   0.027     1.008028    1.145172
------------------------------------------------------------------------------
```

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 and exponential hazard functions. 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 informative 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) connect(direct direct)
```

Figure 11.7

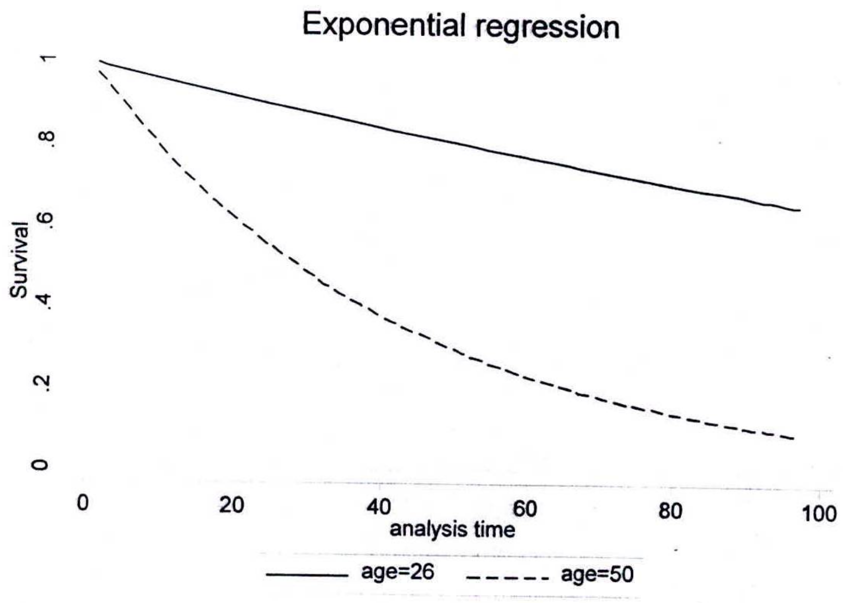


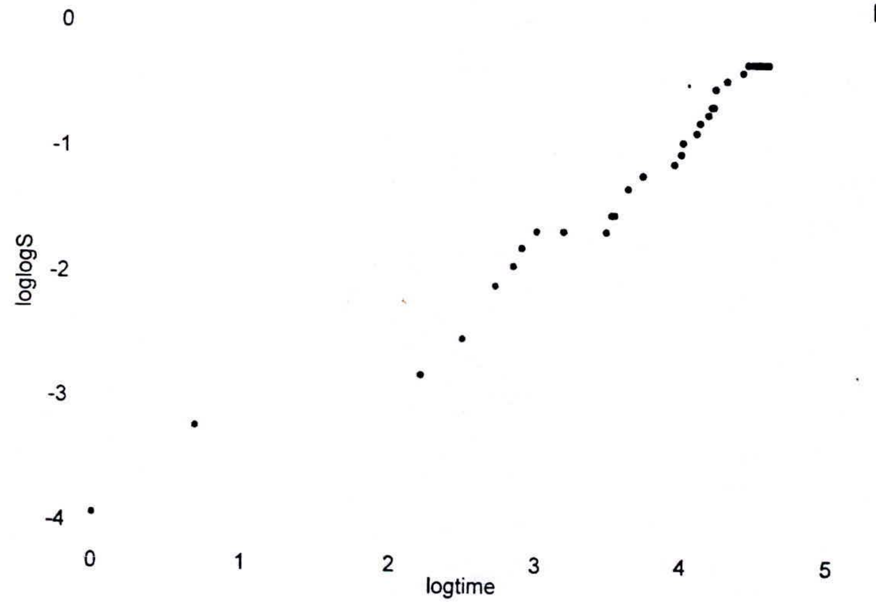
Figure 11.7 shows the predicted survival curve (for transition from HIV diagnosis to AIDS) falling more steeply among older patients. The significant *age* hazard ratio greater than 1 in our exponential regression table implied the same thing, but using **`stcurve`** with **`at1()`** and **`at2()`** 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 11.8. 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 11.8 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 11.8



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
Time at risk    =          3164
                                                 LR chi2(1)      =      4.68
Log likelihood  =    -59.778257                  Prob > chi2     =    0.0306

------------------------------------------------------------------------------
          _t | Haz. Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |   1.079477   .0363509     2.27   0.023     1.010531    1.153127
-------------+----------------------------------------------------------------
       /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 current options.

Poisson Regression

If events occur independently and with constant probability, then counts of events over a given period of time follow a Poisson distribution. Let r_j represent the incidence rate:

$$r_j = \frac{\text{count of events}}{\text{number of times event could have occurred}} \qquad [11.4]$$

The denominator in [11.4] 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 predictor (x) variables:

$$\ln(r_t) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k \qquad [11.5a]$$

Equivalently, the model describes logs of expected event counts:

$$\ln(\textit{expected count}) = \ln(\textit{exposure}) + \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_k x_k \qquad [11.5b]$$

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 × 8 categories of radiation). For each combination, we know the number of deaths and the number of person-years of exposure.

```
Contains data from C:\data\oakridge.dta
  obs:            56                          Radiation (Selvin 1995:474)
 vars:             4                          21 Jul 2005 09:34
 size:           616 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      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(pyears) irr

Poisson regression                                Number of obs   =         56
                                                  LR chi2(1)      =      14.87
                                                  Prob > chi2     =     0.0001
Log likelihood =  -169.7364                       Pseudo R2       =     0.0420

------------------------------------------------------------------------------
      deaths |        IRR   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         rad |   1.236469   .0603551     4.35   0.000     1.123657    1.360606
      pyears |  (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 *pyears*, 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(β) 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 (see equation [10.4]) is only .042.

To perform a goodness-of-fit test, comparing the Poisson model's predictions with the observed counts, use the follow-up command **poisgof** :

```
. poisgof

          Goodness-of-fit chi2  =  254.5475
          Prob > chi2(54)       =    0.0000
```

These goodness-of-fit test results ($\chi^2 = 254.5, P < .00005$) 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 us to reject 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
Log likelihood =   -71.4653                       Pseudo R2       =     0.5966

------------------------------------------------------------------------------
      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
      pyears | (exposure)
------------------------------------------------------------------------------

. poisgof

          Goodness-of-fit chi2  =  58.00534
          Prob > chi2(53)       =    0.2960
```

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 milliseiverts; *rad* = 3 means 20 to 39 milliseiverts; and so forth. An alternative way to include radiation exposure categories in the regression, while watching for nonlinear effects, is as a set of dummy variables. Below we use the **gen()** option of **tabulate** to create 8 dummy variables, *r1* to *r8*, representing each of the 8 values of *rad*.

```
. tabulate rad, gen(r)

  Radiation |
   exposure |
      level |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |          7       12.50       12.50
          2 |          7       12.50       25.00
          3 |          7       12.50       37.50
          4 |          7       12.50       50.00
          5 |          7       12.50       62.50
          6 |          7       12.50       75.00
          7 |          7       12.50       87.50
          8 |          7       12.50      100.00
------------+-----------------------------------
      Total |         56      100.00

. describe
```

```
Contains data from C:\data\oakridge.dta
  obs:            56                          Radiation (Selvin 1995:474)
 vars:            12                          21 Jul 2005 09:34
 size:         1,064 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      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
r1              byte   %8.0g                  rad==         1.0000
r2              byte   %8.0g                  rad==         2.0000
r3              byte   %8.0g                  rad==         3.0000
r4              byte   %8.0g                  rad==         4.0000
r5              byte   %8.0g                  rad==         5.0000
r6              byte   %8.0g                  rad==         6.0000
r7              byte   %8.0g                  rad==         7.0000
r8              byte   %8.0g                  rad==         8.0000
-------------------------------------------------------------------------------
Sorted by:
```

We now include seven of these dummies (omitting one to avoid multicollinearity) as regression predictors. The additional complexity of this dummy-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 (*r7* and *r8*, corresponding to 100 to 119 and 120 or more milliseiverts). At these levels, the incidence rates are about four times higher.

```
. poisson deaths r2-r8 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]
-------------+----------------------------------------------------------------
          r2 |   1.473591    .426898     1.34   0.181     .8351884    2.599975
          r3 |   1.630688   .6659257     1.20   0.231      .732428    3.630587
          r4 |   2.375967   1.088835     1.89   0.059     .9677429    5.833383
          r5 |   .7275113   .7518255    -0.31   0.758     .0961018    5.511957
          r6 |   1.168477    1.20691     0.15   0.880     .1543195    8.847472
          r7 |   4.433729   3.337738     1.98   0.048     1.013863    19.38915
          r8 |    3.89188   1.640978     3.22   0.001     1.703168    8.893267
         age |   1.961907   .1000652    13.21   0.000     1.775267    2.168169
      pyears | (exposure)
------------------------------------------------------------------------------
```

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 r7 = r8

 ( 1)  [deaths]r7 - [deaths]r8 = 0.0

           chi2(  1) =    0.03
         Prob > chi2 =    0.8676
```

Next, generate a new dummy variable *r78*, which equals 1 if either *r7* or *r8* equals 1:

```
. generate r78 = (r7 | r8)
```

Finally, substitute the new predictor for *r7* and *r8* in the regression:

```
. poisson deaths r2-r6 r78 age, irr ex(pyears) nolog

Poisson regression                                Number of obs   =         56
                                                  LR chi2(7)      =     215.41
                                                  Prob > chi2     =     0.0000
Log likelihood = -69.465332                       Pseudo R2       =     0.6079

------------------------------------------------------------------------------
      deaths |        IRR   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          r2 |   1.473602   .4269013     1.34   0.181     .8351949    2.599996
          r3 |   1.630718   .6659381     1.20   0.231     .7324415    3.630655
          r4 |   2.376065    1.08888     1.89   0.059     .9677823    5.833629
          r5 |   .7278387   .7518538    -0.31   0.758     .0961055    5.512165
          r6 |   1.168507   1.206942     0.15   0.880     .1543236    8.847704
         r78 |   3.980326   1.580024     3.48   0.001     1.828214    8.665833
         age |   1.961722    .100043    13.21   0.000     1.775122    2.167937
      pyears | (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

$$g[\mathrm{E}(y)] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k, \qquad y \sim F \qquad [11.6]$$

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:

$$\mathrm{E}(y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k, \qquad y \sim \text{Normal} \qquad [11.7]$$

If $g[\ ]$ is the logit function and y follows a Bernoulli distribution, we have logit regression instead:

$$\mathrm{logit}[\mathrm{E}(y)] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k, \qquad y \sim \text{Bernoulli} \qquad [11.8]$$

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:

$$\ln[\mathrm{E}(y)] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k, \qquad y \sim \text{Poisson} \qquad [11.9]$$

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 offset. 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)
     lnoffset(exposure) eform jknife
```

where **family()** specifies the *y* distribution family, **link()** the link function, and **lnoffset()** an "exposure" variable such as that needed for Poisson regression. The **eform** option asks for regression coefficients in exponentiated form, exp(β) rather than β. Standard errors are estimated through jackknife (**jknife**) 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.
`jknife`	Jackknife estimate of variance.
`jknife1`	One-step jackknife estimate of variance.
`bstrap`	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 (2001).

Chapter 6 began with the simple regression of mean composite SAT scores (*csat*) on per-pupil expenditures (*expense*) of the 50 U.S. states and District of Columbia (*states.dta*):

```
. regress csat expense
```

We could fit the same model and obtain exactly the same estimates with the following command:

```
. glm csat expense, link(identity) family(gaussian)

Iteration 0:   log likelihood = -279.99869

Generalized linear models                          No. of obs      =        51
Optimization     : ML: Newton-Raphson              Residual df     =        49
                                                   Scale param     =  3577.678
Deviance         =  175306.2097                    (1/df) Deviance =  3577.678
Pearson          =  175306.2097                    (1/df) Pearson  =  3577.678

Variance function: V(u) = 1                        [Gaussian]
Link function    : g(u) = u                        [Identity]
Standard errors  : OIM

Log likelihood   = -279.9986936                    AIC             =  11.05877
BIC              =   175298.346

------------------------------------------------------------------------------
        csat |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     expense |  -.0222756   .0060371    -3.69   0.000    -.0341082   -.0104431
       _cons |   1060.732    32.7009    32.44   0.000     996.6399    1124.825
------------------------------------------------------------------------------
```

Because **`link(identity)`** and **`family(gaussian)`** are default options, we could actually have left them out of the previous **`glm`** command.

The **`glm`** command can do more than just duplicate our **`regress`** results, however. For example, we could fit the same OLS model but obtain bootstrap standard errors:

```
. glm csat expense, link(identity) family(gaussian) bstrap

Iteration 0:    log likelihood = -279.99869

Bootstrap iterations (199)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
..................................................   100
..................................................   150
.................................................

Generalized linear models                          No. of obs      =         51
Optimization     : ML: Newton-Raphson              Residual df     =         49
                                                   Scale param     =   4124.656
Deviance         =  175306.2097                    (1/df) Deviance =   3577.678
Pearson          =  175306.2097                    (1/df) Pearson  =   3577.678

Variance function: V(u) = 1                        [Gaussian]
Link function    : g(u) = u                        [Identity]
Standard errors  : Bootstrap

Log likelihood   = -279.9986936                    AIC             =   11.05877
BIC              =   175298.346

------------------------------------------------------------------------------
             |              Bootstrap
        csat |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     expense |  -.0222756   .0039284    -5.67   0.000    -.0299751   -.0145762
       _cons |   1060.732   25.36566    41.82   0.000     1011.017    1110.448
------------------------------------------------------------------------------
```

The bootstrap standard errors reflect observed variation among coefficients estimated from 199 samples of $n = 51$ cases each, drawn by random sampling with replacement from the original $n = 51$ 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 first **logistic** regression of Chapter 10. In the following example, we ask for jackknife standard errors and odds ratio or exponential-form (**eform**) coefficients:

```
. glm any date, link(logit) family(bernoulli) eform jknife

Iteration 0:   log likelihood = -12.995268
Iteration 1:   log likelihood = -12.991098
Iteration 2:   log likelihood = -12.991096

Jackknife iterations (23)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
.......................

Generalized linear models                          No. of obs      =        23
Optimization     : ML: Newton-Raphson              Residual df     =        21
                                                   Scale param     =         1
Deviance         =  25.98219269                    (1/df) Deviance =  1.237247
Pearson          =   22.8885488                    (1/df) Pearson  =  1.089931

Variance function: V(u) = u*(1-u)                  [Bernoulli]
Link function    : g(u) = ln(u/(1-u))              [Logit]
Standard errors  : Jackknife

Log likelihood   = -12.99109634                    AIC             =  1.303574
BIC              =  19.71120426

------------------------------------------------------------------------------
             |             Jackknife
         any | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        date |   1.002093   .0015486     1.35   0.176     .9990623    1.005133
------------------------------------------------------------------------------
```

The final **poisson** regression of the present chapter corresponds to this **glm** model:

```
. glm deaths r2-r6 r78 age, link(log) family(poisson)
      lnoffset(pyears) eform
```

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.

12

Principal Components, Factor, and Cluster Analysis

Principal components 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 often lead to divergent results. Experimenting with these options can tell us how stable a particular finding is, or how much it depends on arbitrary choices about the specific analytical technique.

Stata accomplishes principal components and factor analysis with five basic commands:

`pca`	Principal components analysis.
`factor`	Extracts factors of several different types.
`greigen`	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`**.
`score`	Generates factor scores (composite variables) after **`pca`** , **`factor`** , or **`rotate`**.

The composite variables generated by **`score`** can subsequently be saved, listed, graphed, or analyzed like any other Stata variable.

Users who create composite variables by the older method of adding other variables together without doing factor analysis could assess their results by calculating an α 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, and less theoretical, 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.

Methods described in this chapter can be accessed through the following menus:

Statistics – Other multivariate analysis

Graphics – More statistical graphs

Statistics – Cluster analysis

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.

. `greigen`

Graphs eigenvalues versus factor or component number from the most recent **`factor`** command (also known as a "scree graph").

. `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.

. `score 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*-*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, 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 tree, ylabel(0(.5)3) cutnumber(20) vertlabel`

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 dendrogram`** does the same thing as **`cluster tree`**.

. **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 Components

To illustrate basic principal components and factor analysis commands, we will use a small dataset describing the nine major 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.

```
Contains data from C:\data\planets.dta
  obs:             9                          Solar system data
 vars:            12                          22 Jul 2005 09:49
 size:           441 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
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
logdsun         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
```

To extract initial factors or principal components, use the command **factor** followed by a variable list (variables in any order) and one of the following options:

pcf	Principal components factoring
pf	Principal factoring (default)
ipf	Principal factoring with iterated communalities
ml	Maximum-likelihood factoring

Principal components are calculated through the specialized command **pca** . Type **help pca** or **help factor** to see options for these commands.

To obtain principal components factors, type

```
.  factor rings logdsun - logdense, pcf
(obs=9)

            (principal component factors; 2 factors retained)
  Factor     Eigenvalue      Difference     Proportion     Cumulative
------------------------------------------------------------------------
     1         4.62365         3.45469        0.7706          0.7706
     2         1.16896         1.05664        0.1948          0.9654
     3         0.11232         0.05395        0.0187          0.9842
     4         0.05837         0.02174        0.0097          0.9939
     5         0.03663         0.03657        0.0061          1.0000
     6         0.00006               .        0.0000          1.0000

                Factor Loadings
  Variable |      1            2       Uniqueness
-----------+-------------------------------------
     rings |   0.97917      0.07720     0.03526
   logdsun |   0.67105     -0.71093     0.04427
    lograd |   0.92287      0.37357     0.00875
  logmoons |   0.97647      0.00028     0.04651
   logmass |   0.83377      0.54463     0.00821
  logdense |  -0.84511      0.47053     0.06439
```

Only the first two components have eigenvalues greater than 1, and these two components explain over 96% of the six variables' combined variance. The unimportant 3rd through 6th principal components might safely be disregarded in subsequent analysis.

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

The principal components factoring (**`pcf`**) procedure automatically drops factors with eigenvalues below 1, so

```
. 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 **`greigen`** command. A horizontal line at eigenvalue = 1 in Figure 12.1 marks the usual cutoff for retaining principal components, and again emphasizes the unimportance in this example of components 3 through 6.

```
. greigen, yline(1)
```

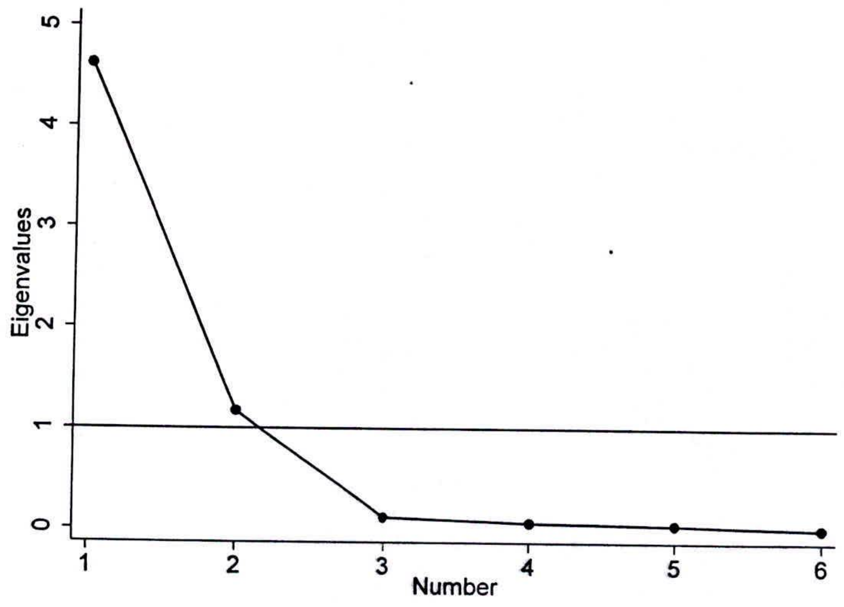


Figure 12.1

Rotation

Rotation further simplifies factor structure. After factoring, type **`rotate`** followed by one of these options:

`varimax`	Varimax orthogonal rotation, for 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 interfactor correlation. **`promax(3)`** is the default.

Two additional **`rotate`** options are

`factors()`	As it does with **`factor`**, this option specifies how many factors to retain.
`horst`	Horst modification to varimax and promax rotation.

Rotation can be performed following any factor analysis, whether it employed the **`pcf`**, **`pf`**, **`ipf`**, or **`ml`** options. In this section, we will follow through on our **`pcf`** example. For orthogonal (default) rotation of the first two components found in the planetary data, we type

```
. rotate

                (varimax rotation)
                   Rotated Factor Loadings
    Variable |        1            2      Uniqueness
-------------+-------------------------------
       rings |    0.52848      0.82792      0.03526
     logdsun |    0.97173      0.10707      0.04427
      lograd |    0.25804      0.96159      0.00875
    logmoons |    0.58824      0.77940      0.04651
     logmass |    0.06784      0.99357      0.00821
    logdense |   -0.88479     -0.39085      0.06439
```

This example 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, with the following command:

```
. rotate, varimax factors(2)
```

For oblique promax rotation (allowing correlated factors) of the most recent factoring, type

```
. rotate, promax

              (promax rotation)
                Rotated Factor Loadings
    Variable |        1            2       Uniqueness
-------------+------------------------------------
       rings |    0.34664      0.76264      0.03526
     logdsun |    1.05196     -0.17270      0.04427
      lograd |    0.00599      0.99262      0.00875
    logmoons |    0.42747      0.69070      0.04651
     logmass |   -0.21543      1.08534      0.00821
    logdense |   -0.87190     -0.16922      0.06439
```

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 2. This appears to be a "large size/many satellites" dimension. *logdsun* and *logdense* load higher on factor 1, forming a "far out/low density" dimension. The next section shows how to create new variables representing these dimensions.

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. **score** performs these calculations automatically, using the most recent **rotate** or **factor** results. In the **score** command we supply names for the new variables, such as *f1* and *f2*.

```
. score f1 f2

              (based on rotated factors)
                 Scoring Coefficients
    Variable |        1            2
-------------+----------------------
       rings |    0.12674      0.22099
     logdsun |    0.48769     -0.09689
      lograd |   -0.03840      0.30608
    logmoons |    0.16664      0.19543
     logmass |   -0.14338      0.34386
    logdense |   -0.39127     -0.01609
```

```
. label variable f1 "Far out/low density"
. label variable f2 "Large size/many satellites"
. list planet f1 f2

     +---------------------------------+
     |  planet          f1          f2 |
     |---------------------------------|
  1. | Mercury   -1.256881   -.9172388 |
  2. |   Venus   -1.188757   -.5160229 |
  3. |   Earth   -1.035242   -.3939372 |
  4. |    Mars   -.5970106   -.6799535 |
  5. | Jupiter    .3841085    1.342658 |
     |---------------------------------|
  6. |  Saturn    .9259058    1.184475 |
  7. |  Uranus    .9347457    .7682409 |
  8. | Neptune    .8161058     .647119 |
  9. |   Pluto    1.017025    -1.43534 |
     +---------------------------------+
```

Being standardized variables, the new factor scores *f1* and *f2* have means (approximately) equal to zero and standard deviations equal to one:

```
. summarize f1 f2

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          f1 |         9    9.93e-09           1  -1.256881   1.017025
          f2 |         9   -3.31e-09           1   -1.43534   1.342658
```

Thus, the factor scores are measured in units of standard deviations from their means. Mercury, for example, is about 1.26 standard deviations below average on the far out/low density (*f1*) dimension because it is actually close to the sun and high density. Mercury is .92 standard deviations below average on the large size/many satellites (*f2*) dimension because it is small and has no satellites. Saturn, in contrast, is .93 and 1.18 standard deviations above average on these two dimensions.

Promax rotation permits correlations between factor scores:

```
. correlate f1 f2
(obs=9)

             |       f1       f2
-------------+------------------
          f1 |   1.0000
          f2 |   0.4974   1.0000
```

Scores on factor 1 have a moderate positive correlation with scores on factor 2: far out/low density planets are more likely also to be larger, with many satellites.

If we employ varimax instead of promax rotation, we get uncorrelated factor scores:

```
. quietly factor rings logdsun - logdense, pcf
. quietly rotate
. quietly score varimax1 varimax2
. correlate varimax1 varimax2
(obs=9)

             | varimax1 varimax2
-------------+------------------
    varimax1 |   1.0000
    varimax2 |   0.0000   1.0000
```

Once created by **score**, factor scores can be treated like any other Stata variable — listed, analyzed, graphed, and so forth. Graphs of principal component factors sometimes help to identify multivariate outliers or clusters of observations that stand apart from the rest. For example, Figure 12.2 reveals three distinct types of planets.

```
. graph twoway scatter f1 f2, yline(0) xline(0) mlabel(planet)
      mlabsize(medsmall) ylabel(, angle(horizontal))
      xlabel(-1.5(.5)1.5, grid)
```

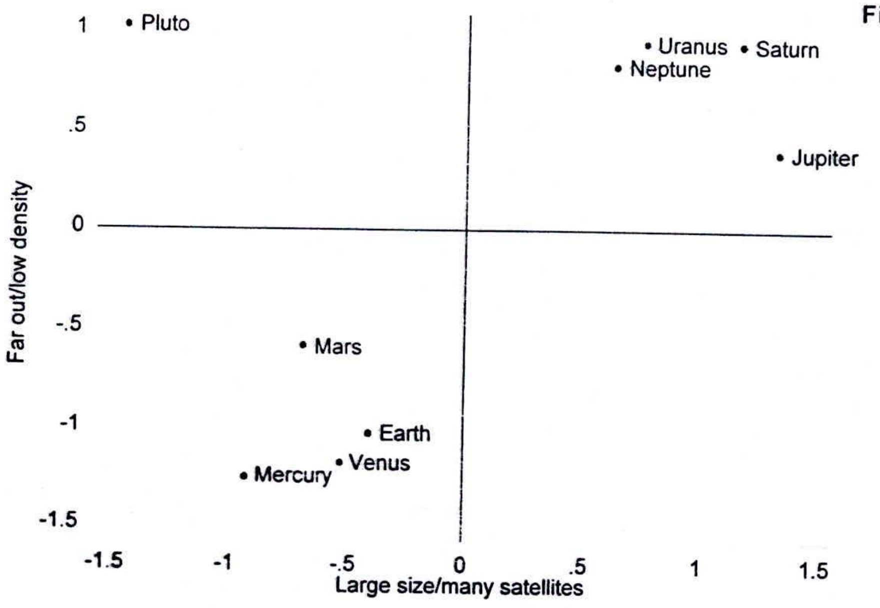


Figure 12.2

The inner, rocky planets (such as Mercury, low on "far out/low density" factor 1; low also on "large size/many satellites" 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 planets for being high on the "far out/low density" dimension, and at the same time low on the "large size/many satellites" dimension.

This example employed rotation. Factor scores obtained by principal components without rotation are often used to analyze large datasets in physical-science fields such as climatology and remote sensing. 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

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 components 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)

             (iterated principal factors; 5 factors retained)
  Factor     Eigenvalue     Difference      Proportion     Cumulative
------------------------------------------------------------------------
     1        4.59663        3.46817         0.7903          0.7903
     2        1.12846        1.05107         0.1940          0.9843
     3        0.07739        0.06438         0.0133          0.9976
     4        0.01301        0.01176         0.0022          0.9998
     5        0.00125        0.00137         0.0002          1.0000
     6       -0.00012           .           -0.0000          1.0000

               Factor Loadings
   Variable |        1          2          3          4          5      Uniqueness
-------------+-------------------------------------------------------------------
      rings |   0.97599    0.16649    0.11374   -0.02065   -0.02234    0.02916
    logdsun |   0.65708   -0.67054    0.14114    0.04471    0.00816    0.09663
     lograd |   0.92670    0.37001   -0.04504    0.04865    0.01662   -0.00036
   logmoons |   0.96738   -0.01074    0.00781   -0.08593    0.01597    0.05636
    logmass |   0.83783    0.54576    0.00557    0.02824   -0.00714   -0.00069
   logdense |  -0.84602    0.48941    0.20594   -0.00610    0.00997    0.00217
```

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)

             (iterated principal factors; 2 factors retained)
  Factor     Eigenvalue     Difference      Proportion     Cumulative
------------------------------------------------------------------------
     1        4.57495        3.47412         0.8061          0.8061
     2        1.10083        1.07631         0.1940          1.0000
     3        0.02452        0.02013         0.0043          1.0043
     4        0.00439        0.00795         0.0008          1.0051
     5       -0.00356        0.02182        -0.0006          1.0045
     6       -0.02537           .           -0.0045          1.0000
```

```
                Factor Loadings
    Variable |        1          2       Uniqueness
-------------+---------------------------------
       rings |    0.97474     0.05374     0.04699
     logdsun |    0.65329    -0.67309     0.12016
      lograd |    0.92816     0.36047     0.00858
    logmoons |    0.96855    -0.02278     0.06139
     logmass |    0.84298     0.54616    -0.00890
    logdense |   -0.82938     0.46490     0.09599
```

After this final factor analysis, we can create composite variables by **rotate** and **score**. Rotation of the **ipf** factors produces results similar to those found earlier with **pcf**: a far out/low density dimension and a large size/many satellites dimension. When variables have a strong factor structure, as these do, the specific techniques we choose make less difference.

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)

              (maximum likelihood factors; 1 factor retained)
  Factor       Variance        Difference      Proportion      Cumulative
-------------------------------------------------------------------------
     1          4.47258             .            1.0000          1.0000

Test:  1 vs. no   factors.  Chi2(   6) =   62.02, Prob > chi2 =   0.0000
Test:  1 vs. more factors.  Chi2(   9) =   51.73, Prob > chi2 =   0.0000

              Factor Loadings
    Variable |        1      Uniqueness
-------------+----------------------
       rings |    0.98726     0.02535
     logdsun |    0.59219     0.64931
      lograd |    0.93654     0.12288
    logmoons |    0.95890     0.08052
     logmass |    0.86918     0.24451
    logdense |   -0.77145     0.40487
```

The **ml** output includes two χ^2 tests:

`J vs. no factors`

This tests whether the current model, with *J* factors, fits the observed correlation matrix significantly better than a no-factor model. A low probability indicates that the current model is a significant improvement over no factors.

`J vs. more factors`

This tests whether the current *J*-factor model fits significantly worse than a more complicated, perfect-fit model. A low *P*-value suggests that the current model does not have enough factors.

The previous 1-factor example yields these results:

`1 vs. no factors`

> χ^2 [6] = 62.02, P = 0.0000 (actually, meaning $P < .00005$). The 1-factor model significantly improves upon a no-factor model.

`1 vs. more factors`

> χ^2 [9] = 51.73, P = 0.0000 ($P < .00005$). The 1-factor model is significantly worse than a perfect-fit model.

Perhaps a 2-factoı model will do better:

```
. factor rings logdsun - logdense, ml nolog factor(2)

(obs=9)

                (maximum likelihood factors; 2 factors retained)
  Factor        Variance         Difference      Proportion     Cumulative
---------------------------------------------------------------------------
     1          3.64200          1.67115          0.6489          0.6489
     2          1.97085               .           0.3511          1.0000

Test:   2 vs. no    factors.  Chi2(  12) =  134.14, Prob > chi2 =  0.0000
Test:   2 vs. more factors.   Chi2(   4) =    6.72, Prob > chi2 =  0.1513

                 Factor Loadings
    Variable |        1           2     Uniqueness
-------------+--------------------------------------
       rings |    0.86551    -0.41545    0.07829
     logdsun |    0.20920    -0.85593    0.22361
     lograd  |    0.98437    -0.17528    0.00028
    logmoons |    0.81560    -0.49982    0.08497
     logmass |    0.99965     0.02639    0.00000
    logdense |   -0.46434     0.88565    0.00000
```

Now we find the following:

`2 vs. no factors`

> χ^2 [12] = 134.14, P = 0.0000 (actually, $P < .00005$). The 2-factor model significantly improves upon a no-factor model.

`2 vs. more factors`

> χ^2 [4] = 6.72, P = 0.1513. The 2-factor model is *not* significantly worse than a perfect-fit model.

These tests suggest that two factors provide an adequate model.

Computational routines performing maximum-likelihood factor analysis often yield "improper solutions" — unrealistic results such as negative variance or zero uniqueness. When this happens (as it did in our 2-factor **`ml`** example), 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 no more than an entry point to begin cluster analysis. We review some basic ideas and illustrate them through a simple example. The following section 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.

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 smaller 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 closest pair of observations between 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 completelinkage` Complete linkage cluster analysis
: Uses the farthest pair of observations between 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 averagelinkage` Average linkage cluster analysis
: Uses the average dissimilarity of observations between 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 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 tree`.

`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 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.

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:

dissimilarity = 1 − similarity

The default dissimilarity measure is the Euclidean distance, option `L2` (or `Euclidean`). This defines the distance between observations *i* and *j* as

$$\{\textstyle\sum_k (x_{ki} - x_{kj})^2\}^{1/2}$$

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`),

$$\textstyle\sum_k (x_{ki} - x_{kj})^2$$

the absolute-value distance (`L1`), maximum-value distance (`Linfinity`), and correlation coefficient similarity measure (`correlation`). Choices for dissimilarities or similarities based on binary variables include simple matching (`matching`), Jaccard binary similarity coefficent (`Jaccard`), and many others. Type `help cldis` for a list and explanations.

Earlier in this chapter, a principal components analysis of variables in *planets.dta* (Figure 12.2) 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 **egen** 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 zlograd = std(lograd)

. egen zlogmoon = std(logmoons)

. egen zlogmass = std(logmass)

. egen zlogdens = std(logdense)

. summ zrings - zlogdens

    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
     zlograd |         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 components 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, 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 tree** command to draw a cluster analysis tree or dendrogram visualizing the most recent hierarchical cluster analysis results (Figure 12.3). The **label(*planet*)** option here causes planet names (values of *planet*) to appear as labels below the tree. Typing **cluster dendrogram** instead of **cluster tree** would produce the same graph.

```
. cluster tree, label(planet) ylabel(0(1)5)
```

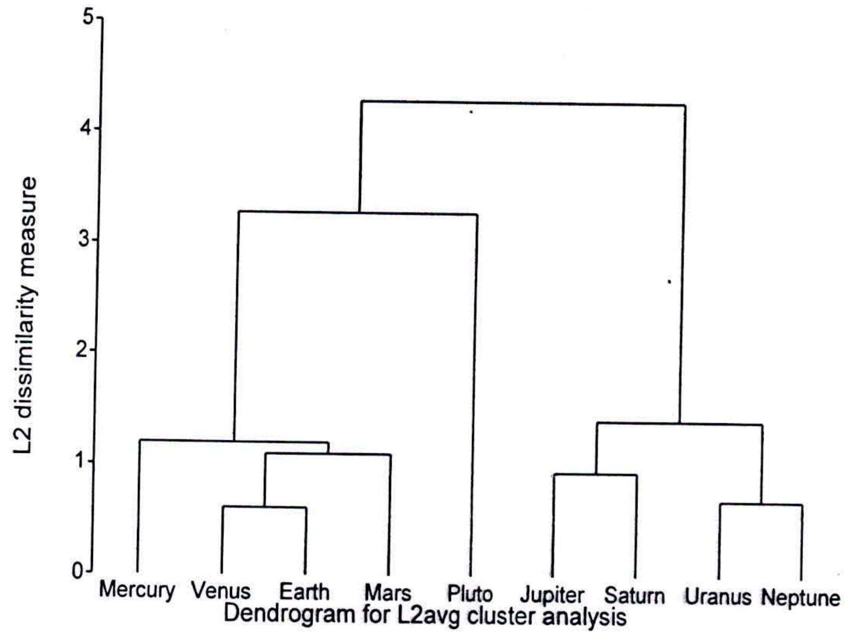


Figure 12.3

Dendrograms such as Figure 12.3 provide key interpretive tools for hierarchical cluster analysis. We can trace the agglomerative process from each observation its own 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 around 1. At this point we have the same three groups suggested in Figure 12.2 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 near 4, the final two clusters fuse.

So, how many types of planets are there? The answer, as Figure 12.3 makes clear, 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 Figure 12.3 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 planet groups (e.g., Mercury–Mars and Earth–Venus) 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, which resembles an outer-system moon more than it does other planets, 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 12.3). 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 nuances of the methods used. The clusters found by one method might not prove replicable under others, or even with slightly different analytical decisions.

Cluster Analysis — 2

Discovering a simple, robust typology to describe the nine planets was straightforward. For a more challenging example, consider the cross-national data in *nations.dta*. This dataset contains living-conditions variables that might provide a basis for classifying countries into types.

```
Contains data from C:\data\nations.dta
  obs:           109                          Data on 109 nations, ca. 1985
 vars:            15                          23 Jul 2005 18:37
 size:         4,142 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
country         str8   %9s                    Country
pop             float  %9.0g                  1985 population in millions
birth           byte   %8.0g                  Crude birth rate/1000 people
death           byte   %8.0g                  Crude death rate/1000 people
chldmort        byte   %8.0g                  Child (1-4 yr) mortality 1985
infmort         int    %8.0g                  Infant (<1 yr) mortality 1985
life            byte   %8.0g                  Life expectancy at birth 1985
```

```
food           int     %8.0g              Per capita daily calories 1985
energy         int     %8.0g              Per cap energy consumed, kg oil
gnpcap         int     %8.0g              Per capita GNP 1985
gnpgro         float   %9.0g              Annual GNP growth % 65-85
urban          byte    %8.0g              % population urban 1985
school1        int     %8.0g              Primary enrollment % age-group
school2        int     %8.0g              Secondary enroll % age-group
school3        byte    %8.0g              Higher ed. enroll % age-group
-------------------------------------------------------------------------------
Sorted by:
```

In Chapter 8, we saw that nonlinear transformations (logs or square roots) 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 our example simple, we will not pursue them here. Linear transformations to standardize the variables in some fashion remain important, however. Otherwise, the variable *gnpcap*, which ranges from about $100 to $19,000 (standard deviation $4,400) would overwhelm other variables such as *life*, which ranges from 40 to 78 years (standard deviation 11 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. The **summarize, detail** command calculates one-variable statistics, and afterwards unobtrusively stores the results in memory as macros (described in Chapter 14). A macro named `r(max)` stores the variable's maximum, and `r(min)` stores its minimum. Thus, to generate new variable *rpop*, defined as a range-standardized version of *pop* (population), type the commands

```
. quietly summ pop, detail
. generate rpop = pop/(r(max) - r(min))
. label variable rpop "Range-standardized population"
```

Similar commands create range-standardized versions of other living-conditions variables:

```
. quietly summ birth, detail
. generate rbirth = birth/(r(max) - r(min))
. label variable rbirth "Range-standardized bith rate"
. quietly summ infmort, detail
. generate rinf = infmort/(r(max) - r(min))
. label variable rinf "Range-standardized infant mortality"
```

and so forth, defining the 8 new variables listed below. These range-standardized variables all have ranges equal to 1.

```
. describe rpop-rschool2

              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
rpop            float  %9.0g                  Range-standardized population
rbirth          float  %9.0g                  Range-standardized bith rate
rinf            float  %9.0g                  Range-standardized infant
                                                mortality
rlife           float  %9.0g                  Range-standardized life
```

```
rfood           float  %9.0g              expectancy
                                         Range-standardized food per
                                           capita
renergy         float  %9.0g             Range-standardized energy per
                                           capita
rgnpcap         float  %9.0g             Range-standardized GNP per
                                           capita
rschool2        float  %9.0g             Range-standardized secondary
                                           school %
```

```
. summarize rpop - rschool2

    Variable |     Obs        Mean    Std. Dev.       Min        Max
-------------+-----------------------------------------------------
        rpop |     109    .0374493    .1206474    .0009622   1.000962
      rbirth |     109    .7452043    .3098672    .2272727   1.227273
        rinf |     109    .4051354    .2913825     .035503   1.035503
       rlife |     109    1.621922     .291343    1.052632   2.052632
       rfood |     108    1.230213    .2644839    .7793776   1.779378
-------------+-----------------------------------------------------
     renergy |     107     .159786    .2137914    .0018464   1.001846
     rgnpcap |     109    .1666459    .2319276    .0057411   1.005741
    rschool2 |     104    .4574849    .2899882    .0196078   1.019608
```

After 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 (**L1**) provides our dissimilarity measure.

```
. cluster waveragelinkage rpop - rschool2, L1 name(L1wav)
```

The full cluster analysis proves unmanageably large for a tree graph:

```
. cluster tree
too many leaves; consider using the cutvalue() or cutnumber() options
r(198);
```

Following the error-message advice, Figure 12.4 employs a **cutnumber(100)** option to form a dendrogram that starts with only 100 groups, after the first few fusions have taken place.

```
. cluster tree, ylabel(0(.5)3) cutnumber(100)
```

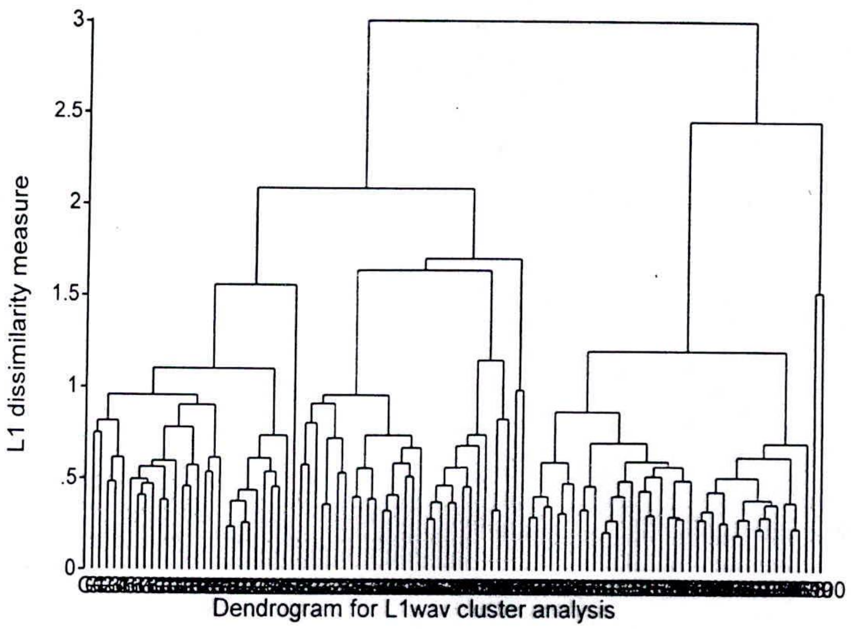


Figure 12.4

The bottom labels in Figure 12.4 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 at far right are unusual; they resist fusion until about 1.5, and then form a stable two-nation group quite different from all the rest. This is one of four clusters remaining above dissimilarities of 2. The first and second of these four final clusters (reading left to right) appear heterogeneous, formed through successive fusion of a number of somewhat distinct major subgroups. The third 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 12.5 gives another view of this analysis, this time using the **`cutvalue(1)`** option to show only clusters with dissimilarities above 1. The **`vertlabel`** option, not really needed here, calls for the bottom labels (G1, G2, etc.) to be printed vertically instead of horizontally.

```
. cluster tree, ylabel(0(.5)3) cutvalue(1) vertlabel
```

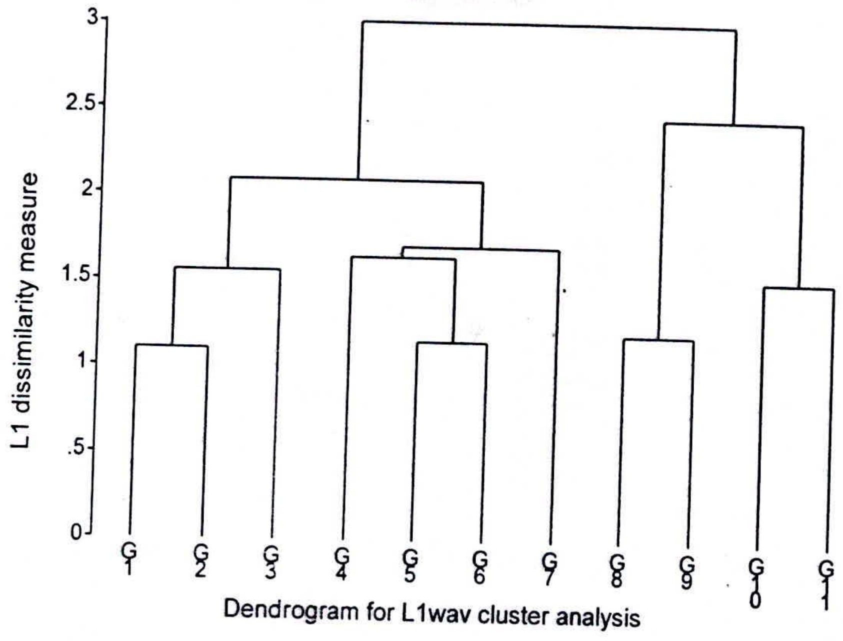


Figure 12.5

As Figure 12.5 shows, there are 11 groups remaining at dissimilarities above 1. 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

```
. by ctype:  list country
```

A more compact list of the same information appears below. This list was produced by copying and pasting data from *nations.dta* into the Data Editor to form a separate, single-purpose dataset in which the columns are country types.

```
     +-------------------------------------------+
     |   ctype1      ctype2     ctype3    ctype4 |
     |-------------------------------------------|
  1. |  Algeria    Argentin   Banglade     China |
  2. |   Brazil    Australi      Benin     India |
  3. |    Burma     Austria    Bolivia           |
  4. |    Chile     Belgium   Botswana           |
  5. | Colombia      Canada   BurkFaso           |
     |-------------------------------------------|
  6. | CostaRic     Denmark    Burundi           |
  7. |   DomRep     Finland   Cameroon           |
  8. |  Ecuador      France   CenAfrRe           |
  9. |    Egypt      Greece   ElSalvad           |
 10. | Indonesi    HongKong   Ethiopia           |
     |-------------------------------------------|
 11. |  Jamaica     Hungary      Ghana           |
 12. |   Jordan     Ireland   Guatemal           |
 13. | Malaysia      Israel     Guinea           |
 14. | Mauritiu       Italy       Haiti   |
 15. |   Mexico       Japan    Honduras   |
     |------------------------------------|
 16. |  Morocco      Kuwait    IvoryCoa   |
 17. |   Panama    Netherla       Kenya   |
 18. | Paraguay    NewZeala     Liberia   |
 19. |     Peru      Norway    Madagasc   |
 20. | Philippi      Poland      Malawi   |
     |------------------------------------|
 21. | SauArabi    Portugal    Mauritan   |
 22. | SriLanka     S_Korea    Mozambiq   |
 23. |    Syria    Singapor       Nepal   |
 24. | Thailand       Spain    Nicaragu   |
 25. |  Tunisia      Sweden       Niger   |
     |------------------------------------|
 26. |   Turkey    TrinToba     Nigeria   |
 27. |  Uruguay         U_K    Pakistan   |
 28. | Venezuel       U_S_A     PapuaNG   |
 29. |             UnArEmir      Rwanda   |
 30. |             W_German     Senegal   |
     |------------------------------------|
 31. |             Yugoslav    SierraLe   |
 32. |                          Somalia   |
 33. |                            Sudan   |
 34. |                         Tanzania   |
 35. |                             Togo   |
     |------------------------------------|
 36. |                          YemenAR   |
 37. |                         YemenPDR   |
 38. |                            Zaire   |
 39. |                           Zambia   |
 40. |                         Zimbabwe   |
     +------------------------------------+
```

The two-nation cluster seen at far right in Figure 12.4 turns out to be type 4, China and India. The broad, homogeneous third cluster in Figure 12.4, type 3, contains a large group of the poorest nations, mainly in Africa. The relatively diverse type 2 contains nations with higher living conditions including the U.S., Europe, and Japan. Type 1, also diverse, contains nations with intermediate conditions. Whether this or some other typology is meaningful remains a substantive question, not a statistical one, and depends on the uses for which a 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.

Time Series Analysis

Stata's evolving time series capabilities are covered in the 350-page *Time-Series Reference Manual*. This chapter provides a brief introduction, beginning with two elementary and useful analytical tools: time plots and smoothing. We then move on to illustrate the use of correlograms, ARIMA models, and 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 technical and thorough treatment of time series topics is found in Hamilton (1994). Other sources include Box, Jenkins, and Reinsel (1994), Chatfield (1996), Diggle (1990), Enders (1995), Johnston and DiNardo (1997), 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(3,1,2)
```

Fits a simple ARIMA(3,1,2) model. Possible options include several estimation strategies, linear constraints, and robust estimates of variance.

```
. arima y, arima(3,1,2) sarima(1,0,1,12)
```

Fits ARIMA model including a multiplicative seasonal component with period 12.

```
. arima D.y x1 L1.x1 x2, ar(1) ma(1 12)
```

Regresses first differences of *y* 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.

```
. dwstat
```
After `regress` , 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.

```
. prais y x1 x2
```
Performs Prais–Winsten regression of *y* on *x1* and *x2*, correcting for first-order autoregressive errors. `prais y x1 x2, corc` does Cochrane–Orcutt instead.

```
. 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(%d)
```
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, then repeats on the residuals. **`tssmooth nl`**, unlike other **`tssmooth`** procedures, cannot work around missing values.

- **`wntestq`** ***`y,`*** **`lags(15)`**

  Box–Pierce 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 (or include a **`generate(`*`newvar`*`)`** option to store the correlations as a variable).

Smoothing

Many time series exhibit rapid up-and-down fluctuations 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:

data = smooth + rough

Dataset *MILwater.dta* contains data on daily water consumption for the town of Milford, New Hampshire over seven months from January through July 1983 (Hamilton 1985b).

```
Contains data from MILwater.dta
  obs:           212                          Milford daily water use, 1 1/83
                                                - 7/31/83
 vars:             4                          27 Jul 2005 12:41
 size:         2,120 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable 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 an arbitrary default. 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 **%d** (daily) display option for this variable.

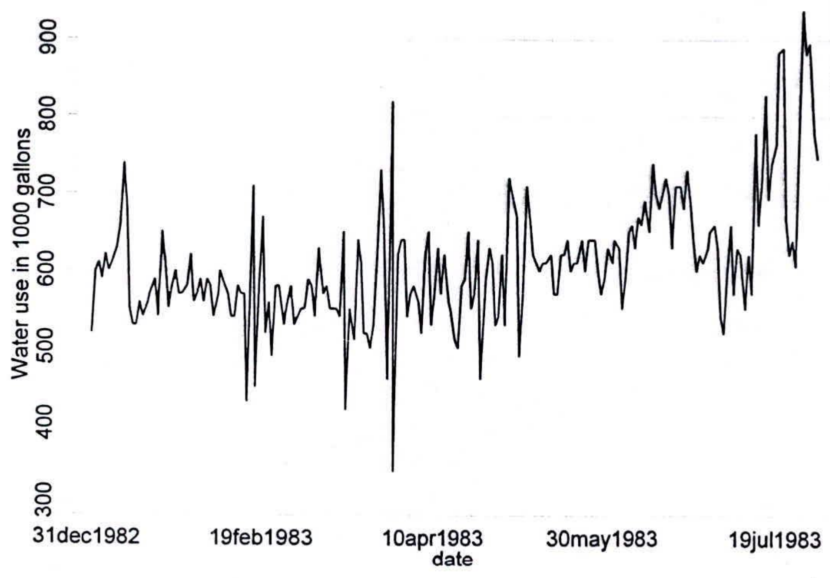
```
. tsset date, format(%d)
        time variable:  date, 01jan1983 to 31jul1983

. 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 numerical values such as "8405" (days since January 1, 1960). If desired, we could use **%d** 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**.

The labeled values of *date* appear in a graph of *water* against *date*, which shows day-to-day variation, as well as an upward trend in water use as summer arrives (Figure 13.1):

```
. graph twoway line water date, ylabel(300(100)900)
```

Figure 13.1



Visual inspection plays an important role in time series analysis. It often helps us to see underlying patterns in jagged series if we smooth the data by calculating a "moving average" at each point from its present, earlier, and later values. 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 water3 = (water[_n-1] + water[_n] + water[_n+1])/3
```

Or, we could apply the **ma** (moving average) function of **egen** :

```
. egen water3 = ma(water), nomiss t(3)
```

The **nomiss** option asks for shorter, uncentered moving averages in the tails; otherwise, the first and last values of *water3* would be missing. The **t(3)** option calls for moving averages of span 3. Any odd-number span ≥3 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 dexponential	double exponential filters
tssmooth hwinters	nonseasonal Holt–Winters smoothing
tssmooth shwinters	seasonal Holt–Winters smoothing
tssmooth nl	nonlinear filters

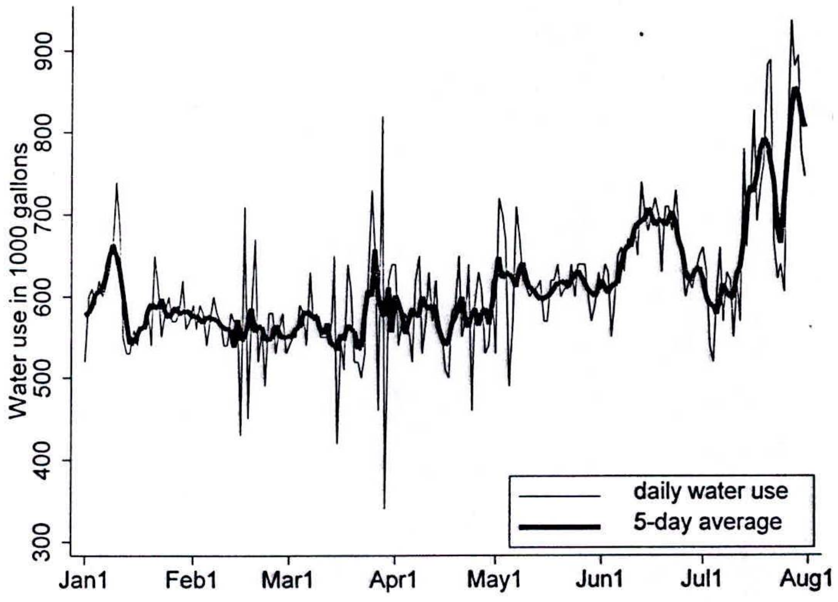
Type **help tssmooth_exponential** , **help tssmooth_hwinters** , etc. for the syntax of each command.

Figure 13.2 graphs a simple 5-day moving average of Milford water use (*water5*), together with the raw data (*water*). This **graph twoway** command overlays a line plot of smoothed *water5* values with a line plot of raw *water* values (thinner line). *X*-axis labels mark start-of-month values chosen "by hand" (8401, 8432, etc.) to make the graph more readable. Readability is also improved by formatting the labels as **%dmd** (date format, but only month followed by day). Compare Figure 13.2's labels with their default counterparts in Figure 13.1.

```
. 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
```

```
. graph twoway line water5 date, clwidth(thick)
      ||  line water date, clwidth(thin) clpattern(solid)
      ||  , ylabel(300(100)900)
      xlabel(8401 8432 8460 8491 8521 8552 8582 8613,
         grid format(%dmd))
      xtitle("") ytitle(Water use in 1000 gallons)
      legend(order(2 1) position(4) ring(0) rows(2)
         label(1 "5-day average") label(2 "daily water use"))
```

Figure 13.2



Moving averages share a drawback of other mean-based statistics: they have little resistance to outliers. Because outliers form prominent spikes in Figure 13.1, we might also try a different smoothing approach. The **tssmooth nl** command performs outlier-resistant nonlinear smoothing, employing methods and a 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" (¼, ½, and ¼ -weighted moving averages of span 3) and other techniques, can be specified in Velleman's original notation. One compound smoother that seems particularly useful is called "4253h, twice." Applying this to *water*, we calculate smoothed variable *water4r*:

```
. tssmooth nl water4r = water, smoother(4253h,twice)
```

Figure 13.3 graphs new smoothed values, *water4r*. Compare Figure 13.3 with 13.2 to see how the 4253h, twice smoothing performs relative to a moving-average. Although both smoothers have similar spans, 4253h, twice does more to reduce the jagged variations.

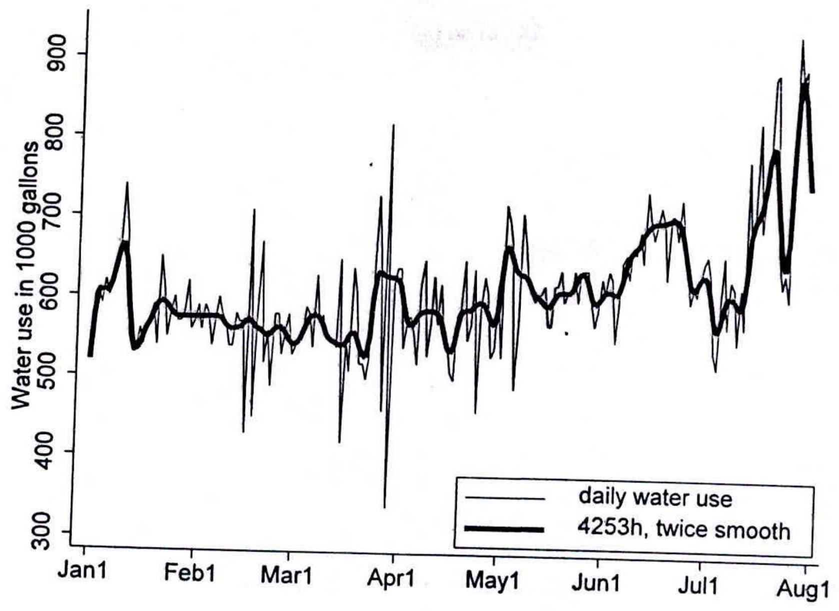


Figure 13.3

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 the results in another time plot, Figure 13.4.

```
. generate rough = water - water4r
. label variable rough "Residuals from 4253h, twice"
. graph twoway line rough date,
        xlabel(8401 8432 8460 8491 8521 8552 8582 8613,
           grid format(%dmd)) xtitle("")
```

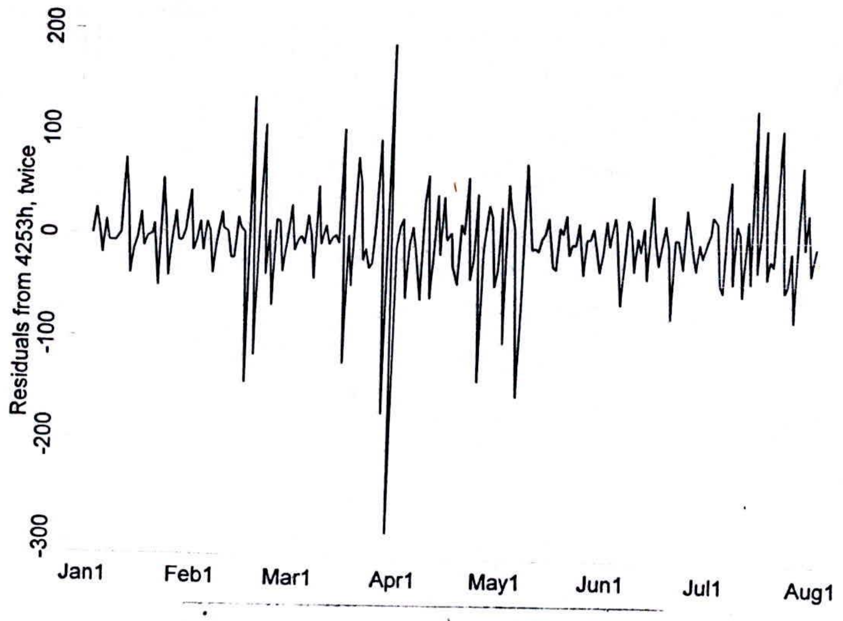


Figure 13.4

The wildest fluctuations in Figure 13.4 occur around March 27–29. Water use abruptly dropped, rose again, and then dropped even further before returning to 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, but they were reassured after the questionable well was taken offline.

The smoothing techniques described in this section tend to make the most sense when the observations are equally spaced in time. For time series with uneven spacing, lowess regression (see Chapter 8) provides a practical alternative.

Further Time Plot Examples

Dataset *atlantic.dta* contains time series of climate, ocean, and fisheries variables for the northern Atlantic from 1950–2000 (the original data sources include Buch 2000, and others cited in Hamilton, Brown, and Rasmussen 2003). The variables include sea temperatures on Fylla Bank off west Greenland; air temperatures in Nuuk, Greenland's capital city; two climate indexes called the North Atlantic Oscillation (NAO) and the Arctic Oscillation (AO); and catches of cod and shrimp in west Greenland waters.

```
Contains data from atlantic.dta
  obs:            51                          Greenland climate & fisheries
 vars:             8                          27 Jul 2005 12:41
 size:         1,734 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
year            int    %ty                    Year
fylltemp        float  %9.0g                  Fylla Bank temp. at 0-40m
fyllsal         float  %9.0g                  Fylla Bank salinity at 0-40m
nuuktemp        float  %9.0g                  Nuuk air temperature
wNAO            float  %9.0g                  Winter (Dec-Mar)
                                                Lisbon-Stykkisholmur NAO
wAO             float  %9.0g                  Winter (Dec-Mar) AO index
tcod1           float  %9.0g                  Division 1 cod catch, 1000t
tshrimp1        float  %9.0g                  Division 1 shrimp catch, 1000t
-------------------------------------------------------------------------------
Sorted by:  year
```

Before analyzing these time series, we **tsset** the dataset, which tells Stata that the variable *year* contains the time-sequence information.

```
. tsset year, yearly

        time variable:  year, 1950 to 2000
```

With a **tsset** dataset, two new qualifiers become available: **tin** (times **in**) and **twithin** (times **within**). To list Fylla temperatures and NAO values for the years 1950 through 1955, type

```
. list year fylltemp wNAO if tin(1950,1955)

     +-------------------------+
     | year   fylltemp    wNAO |
     |-------------------------|
  1. | 1950        2.1     1.4 |
  2. | 1951        1.9   -1.26 |
  3. | 1952        1.6     .83 |
 4. | 1953          2.1        .18 |
 5. | 1954          2.3        .13 |
    |--------------------------|
 6. | 1955          1.2      -2.52 |
    +--------------------------+
```

The **twithin** qualifier works similarly, but excludes the two endpoints:

```
. list year fylltemp wNAO if twithin(1950,1955)
```

```
   +--------------------------+
   | year    fylltemp    wNAO |
   |--------------------------|
2. | 1951         1.9   -1.26 |
3. | 1952         1.6     .83 |
4. | 1953         2.1     .18 |
5. | 1954         2.3     .13 |
   +--------------------------+
```

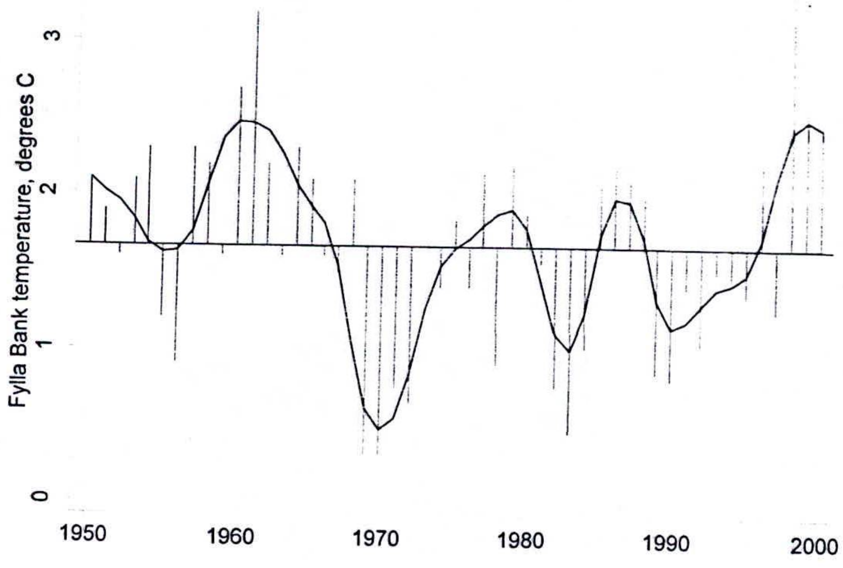
We use **tssmooth nl** to define a new variable, *fyll4*, containing 4253h, twice smoothed values of *fylltemp* (data from Buch 2000).

```
. tssmooth nl fyll4 = fylltemp, smoother(4253h, twice)
```

Figure 13.5 graphs raw (*fylltemp*) and smoothed (*fyll4*) Fylla Bank temperatures. Raw temperatures are shown as spike-plot deviations from the mean (1.67 °C), so this graph emphasizes both decadal cycles and annual variations.

```
. graph twoway spike fylltemp year, base(1.67) yline(1.67)
      || line fyll4 year, clpattern(solid)
      || , ytitle("Fylla Bank temperature, degrees C") ylabel(0(1)3)
     xtitle("") xtick(1955(10)1995) legend(off)
```

Figure 13.5



The smoothed values of Figure 13.5 exhibit irregular periods of generally warmer and cooler water. Of course, "warmer" is a relative term around Greenland; these summer sea temperatures rise no higher than 3.34 °C (37 °F).

Fylla Bank temperatures are influenced by a large-scale atmospheric pattern called the North Atlantic Oscillation, or NAO. Figure 13.6 graphs smoothed temperatures together with smoothed values of the NAO (a new variable named *wNAO4*). For this overlaid graph, temperature defines the left axis scale, **yaxis(1)**, and NAO the right, **yaxis(2)**. Further *y*-axis options specify whether they refer to axis 1 or 2. For example, a horizontal line drawn by **yline(0, axis(2))** marks the zero point of the NAO index. On both axes, numerical labels are written horizontally. The legend appears at the 5 o'clock position inside the plot space, **position(5) ring(0)**.

```
. graph twoway line fyll4 year, yaxis(1)
     ylabel(0(1)3, angle(horizontal) nogrid axis(1))
     ytitle("Fylla Bank temperature, degrees C", axis(1))
     || line wNAO4 year, yaxis(2) ytitle("Winter NAO index", axis(2))
     ylabel(-3(1)3, angle(horizontal) axis(2)) yline(0, axis(2))
     ||  , xtitle("") xlabel(1950(10)2000, grid) xtick(1955(5)1995)
     legend(label(1 "Fylla temperature") label(2 "NAO index") cols(1)
        position(5) ring(0))
```

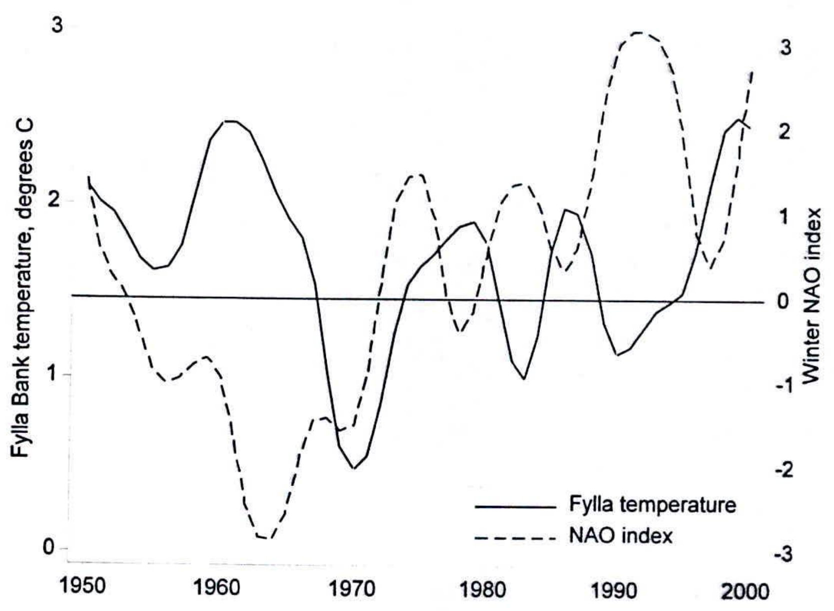


Figure 13.6

Overlaid plots provide a way to visually examine how several time series vary together. In Figure 13.6, we see evidence of a negative correlation: high-NAO periods correspond to low temperatures. The physical mechanism behind this correlation involves northerly winds that bring Arctic air and water to west Greenland during high-NAO phases. The negative temperature–NAO correlation became stronger during the later part of this time series, roughly the years 1973 to 1997. We will return to this relationship in later sections.

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 *wNAO_1*, equal to the previous year's NAO value:

```
. generate wNAO_1 = wNAO[_n-1]
(1 missing value generated)
```

An alternative way to achieve the same thing, using **tsset** data, is with Stata's **L.** (lag) operator:

```
. generate wNAO_1 = L.wNAO
(1 missing values generated)
```

Lag operators are often simpler than an explicit-subscripting approach. More importantly, the lag operators also respect panel data. To generate lag 2 values, use

```
. generate wNAO_2 = L2.wNAO
(2 missing values generated)

. list year wNAO wNAO_1 wNAO_2 if tin(1950,1954)

     +----------------------------------+
     | year    wNAO    wNAO_1    wNAO_2 |
     |----------------------------------|
  1. | 1950     1.4         .         . |
  2. | 1951   -1.26       1.4         . |
  3. | 1952     .83     -1.26       1.4 |
  4. | 1953     .18       .83     -1.26 |
  5. | 1954     .13       .18       .83 |
     +----------------------------------+
```

We could have obtained this same list without generating any new variables, by instead typing

```
. list year wNAO L.wNAO L2.wNAO if tin(1950,1954)
```

The **L.** operator is one of several that simplify the analysis of **tsset** datasets. Other time series operators are **F.** (lead), **D.** (difference), and **S.** (seasonal difference). These operators can be typed in upper or lowercase — for example, **F2.wNAO** or **f2.wNAO**.

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 $y_t - y_{t-1}$, (which is the same as **D.**)
S2.	Second seasonal difference $(y_t - y_{t-2})$ (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 monthly temperatures instead of yearly, we might want to calculate `S12.temp`, which would be the differences between December 2000 temperature and December 1999 temperature, November 2000 temperatures and November 1999 temperature, and so forth.

Lag operators can appear directly in most analytical commands. We could regress 1973–97 *fylltemp* on *wNAO*, including as additional predictors *wNAO* values from one, two, and three years previously, without first creating any new lagged variables.

```
. regress fylltemp wNAO L1.wNAO L2.wNAO L3.wNAO if tin(1973,1997)

      Source |       SS       df       MS              Number of obs =      25
-------------+------------------------------           F(  4,    20) =    4.57
       Model |  3.1884913     4  .797122826            Prob > F      =  0.0088
    Residual |  3.48929123    20  .174464562           R-squared     =  0.4775
-------------+------------------------------           Adj R-squared =  0.3730
       Total |  6.67778254    24  .278240939           Root MSE      =  .41769

------------------------------------------------------------------------------
fylltemp     |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
wNAO         |
          -- |  -.1688424   .0412995    -4.09   0.001    -.2549917   -.0826931
          L1 |   .0043805   .0421436     0.10   0.918    -.0835294    .0922905
          L2 |  -.0472993    .050851    -0.93   0.363    -.1533725     .058774
          L3 |   .0264682   .0495416     0.53   0.599    -.0768738    .1298102
_cons        |   1.727913   .1213588    14.24   0.000     1.474763    1.981063
-------------+----------------------------------------------------------------
```

Equivalently, we could have typed

```
. regress fylltemp L(0/3).wNAO if tin(1973,1997)
```

The estimated model is

$$\text{predicted}\,\textit{fylltemp}_t = 1.728 - .169wNAO_t + .004wNAO_{t-1} - .047wNAO_{t-2} + .026wNAO_{t-3}$$

Coefficients on the lagged terms are not statistically significant; it appears that current (unlagged) values of $wNAO_t$ provide the most parsimonious prediction. Indeed, if we re-estimate this model without the lagged terms, the adjusted R^2 rises from .37 to .43. Either model is very rough, however. A Durbin–Watson test for autocorrelated errors is inconclusive, but that is not reassuring given the small sample size.

```
. dwstat

Durbin-Watson d-statistic(  5,    25) =  1.423806
```

Autocorrelated errors, commonly encountered with time series, invalidate the usual OLS confidence intervals and tests. More suitable regression methods for time series are discussed later in this chapter.

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 Cor[y_t, y_{t-2}], and so forth. A correlogram graphs correlation versus lags.

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 fylltemp, lags(9)
```

```
                                          -1       0       1 -1       0       1
 LAG       AC       PAC      Q     Prob>Q  [Autocorrelation]  [Partial Autocor]
-------------------------------------------------------------------------------
1        0.4038   0.4141   8.8151  0.0030          |---               |---
2        0.1996   0.0565   11.012  0.0041          |-                 |
3        0.0788   0.0045   11.361  0.0099          |                  |
4        0.0071  -0.0556   11.364  0.0228          |                  |
5       -0.1623  -0.2232   12.912  0.0242         -|                  |
6       -0.0733   0.0880   13.234  0.0395          |                 -|
7        0.0490   0.1367   13.382  0.0633          |                  |
8       -0.1029  -0.2510   14.047  0.0805          |                  |-
9       -0.2228  -0.2779   17.243  0.0450         -|                --|
                                                                    --|
```

Lags appear at the left side of the table, and are followed by columns for the autocorrelations (AC) and partial autocorrelations (PAC). For example, the correlation between *fylltemp*$_t$ and *fylltemp*$_{t-2}$ is .1996, and the partial autocorrelation (adjusted for lag 1) is .0565. The Q statistics (Box–Pierce portmanteau) test a series of null hypotheses that all autocorrelations up to and including each lag are zero. Because the P-values seen here are mostly below .05, we can reject the null hypothesis, and conclude that *fylltemp* shows significant autocorrelation. If none of the Q statistics had been below .05, we might conclude instead that the series was "white noise" with no significant autocorrelation.

At the right in this output are character-based plots of the autocorrelations and partial autocorrelations. Inspection of such plots plays a role in the specification of time series models. More refined graphical autocorrelation plots can be obtained through the **ac** command:

```
. ac fylltemp, lags(9)
```

The resulting correlogram, Figure 13.7, includes a shaded area marking pointwise 95% confidence intervals. Correlations outside of these intervals are individually significant.

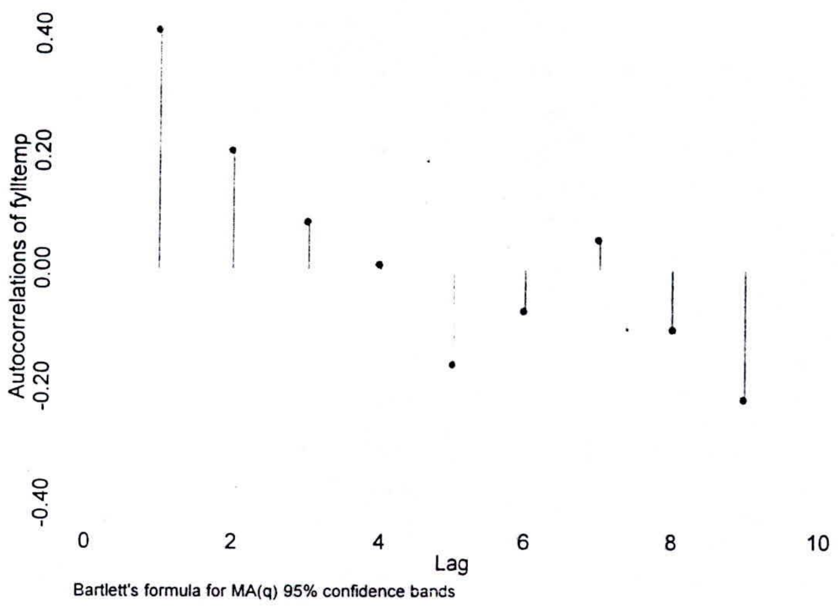


Figure 13.7

A similar command, **pac**, produces the graph of partial autocorrelations seen in Figure 13.8. Approximate confidence intervals (estimating the standard error as $1/\sqrt{n}$) also appear in Figure 13.8. The default plot produced by both **ac** and **pac** has the look shown in Figure 13.7. For Figure 13.8 we chose different options, drawing a baseline at zero correlation, and indicating the confidence interval as an outline instead of a shaded area.

```
. pac fylltemp, yline(0) lags(9) ciopts(bstyle(outline))
```

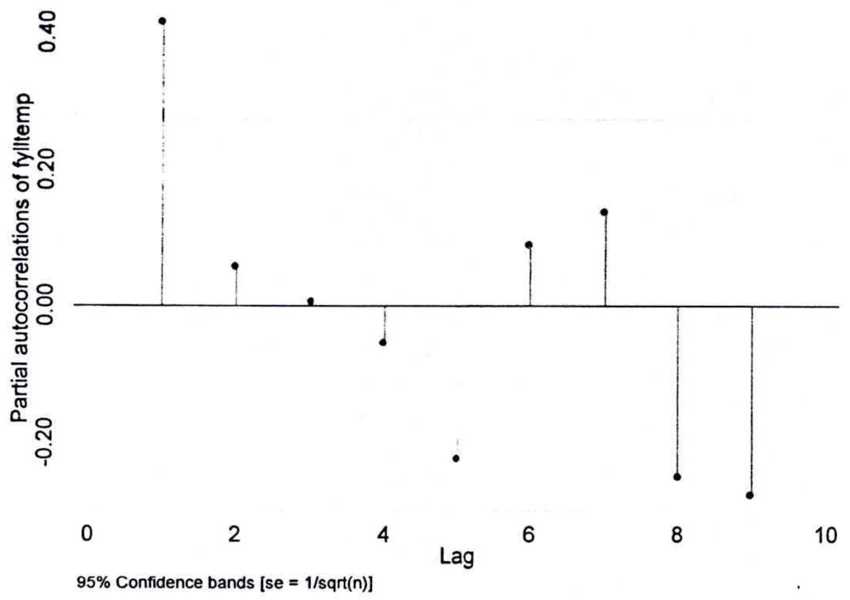


Figure 13.8

Cross-correlograms help to explore relationships between two time series. Figure 13.9 shows the cross-correlogram of *wNAO* and *fylltemp* over 1973–97. The cross-correlation is substantial and negative at 0 lag, but is closer to zero at other positive or negative lags. This suggests that the relationship between the two series is "instantaneous" (in yearly data) rather than delayed or distributed over several years. Recall the nonsignificance of lagged predictors from our earlier OLS regression.

```
. xcorr wNAO fylltemp if tin(1973,1997), lags(9) xlabel(-9(1)9, grid)
```

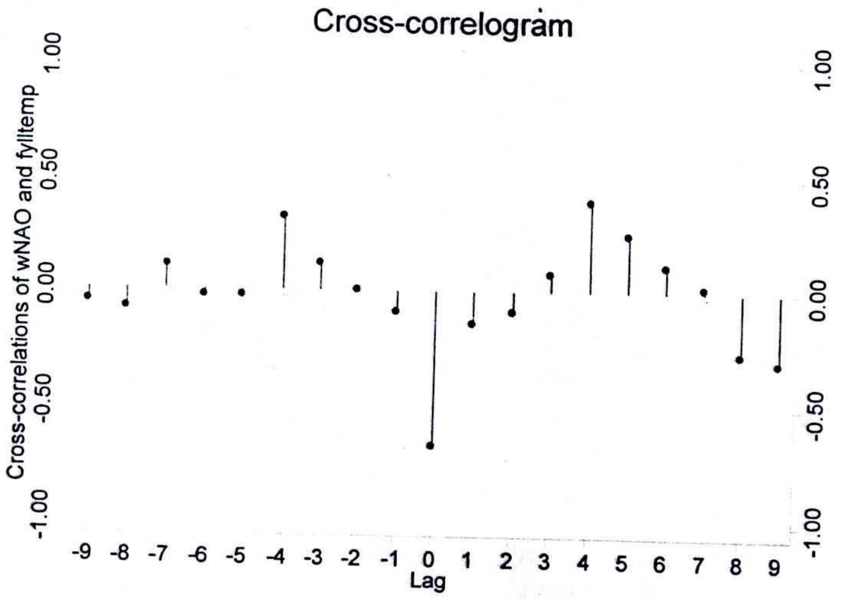


Figure 13.9

If we list our input or independent variable first in the **`xcorr`** command, and the output or dependent variable second — as was done for Figure 13.9 — then positive lags denote correlations between input at time *t* and output at time *t* +1, *t* +2, etc. Thus, we see a positive correlation of .394 between winter NAO index and Fylla temperature four years later.

The actual cross-correlation coefficients, and a text version of the cross-correlogram, can be obtained with the **`table`** option:

```
. xcorr wNAO fylltemp if tin(1973,1997), lags(9) table

                           -1       0       1
 LAG          CORR         [Cross-correlation]
-----------------------------------------------
-9             -0.0541              |
-8             -0.0786              |
-7              0.1040              |
-6             -0.0261              |
-5             -0.0230              |
-4              0.3185              |--
-3              0.1212              |
-2              0.0053              |
-1             -0.0909              |
0              -0.6740         -----|
1              -0.1386             -|
2              -0.0865              |
3          0.0757               |
4          0.3940               |---
5          0.2464               |-
6          0.1100               |
7          0.0183               |
8         -0.2699             --|
9         -0.3042             --|
```

ARIMA Models

Autoregressive integrated moving average (ARIMA) models for time series can be estimated through the **arima** command. This command encompasses simple autoregressive (AR), moving average (MA), or ARIMA models of any order. It also can estimate structural models that include one or more predictor variables and AR or MA errors. The general form of such structural models, in matrix notation, is

$$y_t = \mathbf{x}_t \beta + \mu_t \qquad [13.1]$$

where y_t is the vector of dependent-variable values at time t, $\mathbf{x}_t$ is a matrix of predictor-variable values (usually including a constant), and μ_t is a vector of disturbances. Those disturbances can be autoregressive or moving-average, of any order. For example, ARMA(1,1) disturbances are

$$\mu_t = \rho\mu_{t-1} + \theta\epsilon_{t-1} + \epsilon_t \qquad [13.2]$$

where ρ is the first-order autocorrelation parameter, θ is the first-order moving average parameter, and ϵ is a white-noise (normal i.i.d.) disturbance. **arima** fits simple models as a special case of [13.1] and [13.2], with a constant (β_0) replacing the structural term $\mathbf{x}_t\beta$. Therefore, a simple ARMA(1,1) model becomes

$$\begin{aligned} y_t &= \beta_0 + \mu_t \\ &= \beta_0 + \rho\mu_{t-1} + \theta\epsilon_{t-1} + \epsilon_t \end{aligned} \qquad [13.3]$$

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:

$$y_t = \alpha + \rho y_{t-1} + \theta\epsilon_{t-1} + \epsilon_t \qquad [13.4]$$

Because in the simple structural model $y_t = \beta_0 + \mu_t$, equation [13.3] (Stata's version) is equivalent to [13.4], apart from rescaling the constant $\alpha = (1-\rho)\beta_0$.

Using **arima**, an ARMA(1,1) model (equation [13.3]) 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)
```

Either command 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 disturbances (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(1,0,1) errors, an appropriate command would be

```
. arima y x L.x w, arima(1,0,1)
```

Although seasonal differencing (e.g., **`S12.y`**) and/or seasonal lags (e.g., **`L12.x`**) can be included, as of this writing **`arima`** does not estimate multiplicative ARIMA$(p,d,q)(P,D,Q)_S$ seasonal models.

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. Formal statistical tests for "unit roots" (a nonstationary AR(1) process in which $\rho_1 = 1$, also known as a "random walk") also help. Stata offers three unit root tests, **`pperron`** (Phillips–Perron), **`dfuller`** (augmented Dickey–Fuller), and **`dfgls`** (augmented Dickey–Fuller using GLS, generally a more powerful test than **`dfuller`**).

Applied to Fylla Bank temperatures, a **`pperron`** test rejects the null hypothesis of a unit root ($P < .01$).

```
. pperron fylltemp, lag(3)

Phillips-Perron test for unit root                  Number of obs    =        50
                                                    Newey-West lags  =         3

                                ---------- Interpolated Dickey-Fuller ---------
                  Test          1% Critical       5% Critical      10% Critical
               Statistic           Value             Value             Value
-------------------------------------------------------------------------------
 Z(rho)         -29.871           -18.900           -13.300           -10.700
 Z(t)            -4.440            -3.580            -2.930            -2.600
-------------------------------------------------------------------------------
* MacKinnon approximate p-value for Z(t) = 0.0003
```

Similarly, a Dickey–Fuller GLS test evaluating the null hypothesis that *fylltemp* has a unit route (versus the alternative hypothesis that it is stationary with a possibly nonzero mean. but no linear time trend) rejects this null hypothesis ($P < .05$). Both tests thus confirm the visual impression of stationarity given by Figure 13.5.

```
. dfgls fylltemp, notrend maxlag(3)

DF-GLS for fylltemp                    Number of obs =    47

               DF-GLS mu          1% Critical        5% Critical       10% Critical
  [lags]     Test Statistic          Value              Value             Value
------------------------------------------------------------------------------------
     3          -2.304              -2.620             -2.211            -1.913
     2          -2.479              -2.620             -2.238            -1.938
     1          -3.008              -2.620             -2.261            -1.959

Opt Lag (Ng-Perron seq t) = 0 [use maxlag(0)]
Min SC   = -.6735952 at lag  1 with RMSE  .6578912
Min MAIC = -.2683716 at lag  2 with RMSE  .6569351
```

For a stationary series, 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, 36 in monthly data) indicate a seasonal pattern. Identification of seasonal models follows similar guidelines, but applied to autocorrelations and partial autocorrelations at seasonal lags.

Figures 13.7–13.8 weakly suggest an AR(1) process, so we will try this as a simple model for *fylltemp*.

```
. arima fylltemp, arima(1,0,0) nolog

ARIMA regression

Sample:  1950 to 2000                           Number of obs      =        51
                                                Wald chi2(1)       =      7.53
Log likelihood = -48.66274                      Prob > chi2        =    0.0061
------------------------------------------------------------------------------
             |                 OPG
fylltemp     |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
fylltemp     |
_cons        |    1.68923   .1513096    11.16   0.000     1.392669    1.985792
-------------+----------------------------------------------------------------
ARMA         |
ar           |
          L1 |   .4095759   .1492491     2.74   0.006     .1170531    .7020987
-------------+----------------------------------------------------------------
      /sigma |    .627151   .0601859    10.42   0.000     .5091889    .7451131
------------------------------------------------------------------------------
```

After we fit an **arima** model, its coefficients and other results are saved temporarily in Stata's usual way. For example, to see the recent model's AR(1) coefficient and s.e., type

```
. display [ARMA]_b[L1.ar]
.4095759

. display [ARMA]_se[L1.ar]
.14924909
```

The AR(1) coefficient in this example is statistically distinguishable from zero ($t = 2.74$, $p = .006$), which gives one indication of model adequacy. A second test is whether the residuals appear to be uncorrelated "white noise." We can obtain residuals (also predicted values, and other case statistics) after **arima** through **predict**:

```
. predict fyllres, resid

. corrgram fyllres, lags(15)

                                          -1       0       1 -1       0       1
 LAG       AC       PAC      Q     Prob>Q  [Autocorrelation]  [Partial Autocor]
-------------------------------------------------------------------------------
1      -0.0173  -0.0176    .0162  0.8987
2       0.0467   0.0465   .13631  0.9341                             |
3       0.0386   0.0497   .22029  0.9742                             |
4       0.0413   0.0496   .31851  0.9886                             |
5      -0.1834  -0.2450   2.2955  0.8069       -                     |
6      -0.0498  -0.0602   2.4442  0.8747                            -|
7       0.1532   0.2156   3.8852  0.7929          -                  |
8      -0.0567  -0.0726    4.087  0.8492                             |-
9      -0.2055  -0.3232   6.8055  0.6574       -                     |
10     -0.1156  -0.2418   7.6865  0.6594                           --|
11      0.1397   0.2794   9.0051  0.6214          -                 -|
12     -0.0028   0.1606   9.0057  0.7024                             |--
13      0.1091   0.0647   9.8519  0.7061                             |-
14      0.1014  -0.0547   10.603  0.7169                             |
15     -0.0673  -0.2837   10.943  0.7566                           --|
```

corrgram's Q test finds no significant autocorrelation among residuals out to lag 15. We could obtain exactly the same result by requesting a **wntestq** (white noise test Q statistic) for 15 lags.

```
. wntestq fyllres, lags(15)

Portmanteau test for white noise
---------------------------------------
 Portmanteau (Q) statistic =    10.9435
 Prob > chi2(15)           =     0.7566
```

By these criteria, our AR(1) or ARIMA(1,0,0) model appears adequate. More complicated versions, with MA or higher-order AR terms, do not offer much improvement in fit.

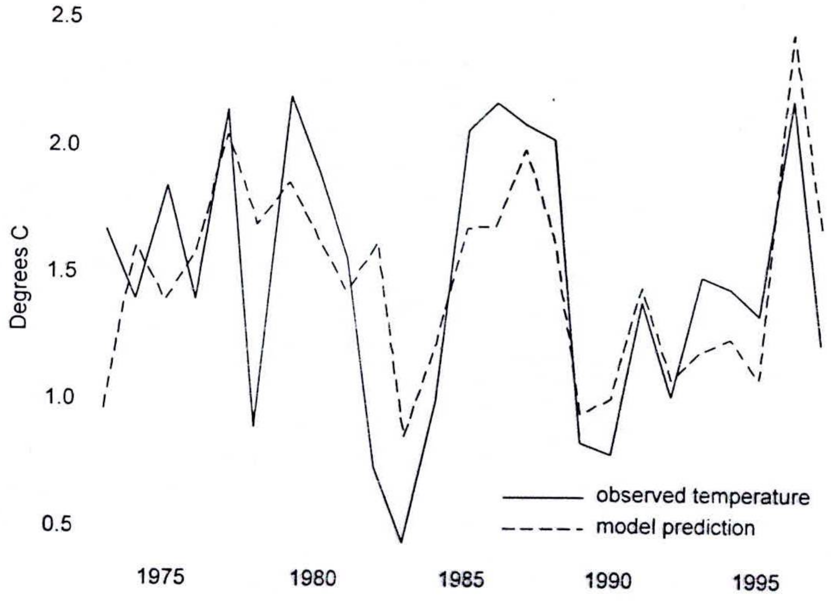
A similar AR(1) model fits *fylltemp* over just the years 1973–1997. During this period, however, information about the winter North Atlantic Oscillation (*wNAO*) significantly improves the predictions. For this model, we include *wNAO* as a predictor but keep an AR(1) term to account for autocorrelation of errors.

```
. arima fylltemp wNAO if tin(1973,1997), ar(1) nolog

ARIMA regression

Sample:  1973 to 1997                          Number of obs      =        25
                                               Wald chi2(2)       =     12.73
Log likelihood = -10.3481                      Prob > chi2        =    0.0017

------------------------------------------------------------------------------
             |                 OPG
fylltemp     |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
fylltemp     |
wNAO         |  -.1736227   .0531688    -3.27   0.001    -.2778317   -.0694138
_cons        |   1.703462   .1348599    12.63   0.000     1.439141    1.967782
-------------+----------------------------------------------------------------
ARMA         |
ar           |
          L1 |   .2965222    .237438     1.25   0.212    -.1688478    .7618921
-------------+----------------------------------------------------------------
      /sigma |     .36536   .0654008     5.59   0.000     .2371767    .4935432
------------------------------------------------------------------------------

. predict fyllhat
(option xb assumed; predicted values)

. label variable fyllhat "predicted temperature"

. predict fyllres2, resid

. corrgram fyllres2, lags(9)

                                          -1       0       1 -1       0       1
LAG       AC       PAC      Q     Prob>Q  [Autocorrelation]  [Partial Autocor]
-------------------------------------------------------------------------------
1       0.1485   0.1529   1.1929  0.2747          |-                 |-
2      -0.1028  -0.1320   1.7762  0.4114          |                 -|
3       0.0495   0.1182   1.9143  0.5904          |                  |
4       0.0887   0.0546   2.3672  0.6686          |                  |
5      -0.1690  -0.2334   4.0447  0.5430         -|                 -|
6      -0.0234   0.0722   4.0776  0.6662          |                  |
7       0.2658   0.3062   8.4168  0.2973          |--                |--
8      -0.0726  -0.2236   8.7484  0.3640          |                 -|
9      -0.1623  -0.0999   10.444  0.3157         -|                  |
```

wNAO has a significant, negative coefficient in this model. The AR(1) coefficient now is not statistically significant. If we dropped the AR term, however, our residuals would no longer pass **`corrgram`**'s test for white noise. Figure 13.10 graphs the predicted values, *fyllhat*, together with the observed temperature series *fylltemp*. The model does reasonably well in fitting the main warming/cooling episodes and a few of the minor variations. To have the *y*-axis labels displayed with the same number of decimal places (0.5, 1.0, 1.5,... instead of .5, 1, 1.5,...) in this graph, we specify their format as **`%2.1f`**.

```
. graph twoway line fylltemp year if tin(1973, 1997)
     || line fyllhat year if tin(1973, 1997)
     || , ylabel(.5(.5)2.5, angle(horizontal) format(%2.1f))
     ytitle("Degrees C") xlabel(1975(5)1995, grid) xtitle("")
     legend(label(1 "observed temperature")
        label(2 "model prediction") position(5) ring(0) col(1))
```

Figure 13.10



A technique called Prais–Winsten regression (**prais**), which corrects for first-order autoregressive errors, can also be illustrated with this example.

```
. prais fylltemp wNAO if tin(1973,1997), nolog

Prais-Winsten AR(1) regression -- iterated estimates

      Source |       SS       df       MS              Number of obs =      25
-------------+------------------------------           F(  1,    23) =   23.14
       Model |  3.35819258     1  3.35819258           Prob > F      =  0.0001
    Residual |  3.33743545    23  .145105889           R-squared     =  0.5016
-------------+------------------------------           Adj R-squared =  0.4799
       Total |  6.69562803    24  .278984501           Root MSE      =  .38093

------------------------------------------------------------------------------
    fylltemp |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        wNAO |   -.17356    .037567     -4.62   0.000    -.2512733   -.0958468
       _cons |  1.703436    .1153695    14.77   0.000     1.464776    1.942096
-------------+----------------------------------------------------------------
         rho |  .2951576
------------------------------------------------------------------------------
Durbin-Watson statistic (original)    1.344998
Durbin-Watson statistic (transformed) 1.789412
```

prais is an older method, more specialized than **arima** . Its regression-based standard errors assume that rho (ρ) is known rather than estimated. Because that assumption is untrue, the standard errors, tests, and confidence intervals given by **`prais`** tend to be anti-conservative, especially in small samples. **`prais`** provides a Durbin–Watson statistic (d = 1.789). In this example, the Durbin–Watson test agrees that after fitting the model, no significant first-order autocorrelation remains.

14

Introduction to Programming

As mentioned in Chapters 2 and 3, 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. Many of the commands used in previous chapters actually involve programs written in Stata. These programs might have originated either from Stata Corporation 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, our ability to write programs in Stata greatly extends what we can do.

Substantial books (*Stata Programming Reference Manual*; *Mata Reference Manual*; *Maximum Likelihood Estimation with Stata*) have been written about Stata programming. This engaging topic is also the focus of periodic NetCourses (see www.stata.com) and a section of the *User's Guide*. The present chapter has the modest aim of introducing a few basic tools and giving examples that show how these tools can be used.

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 ASCII (text) 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 to some other character, through a `#delimit` command. For example,

```
        #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 by typing the following command:

```
        #delimit cr
```

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 commonly go in a directory named C:\ado\personal, whereas the hundreds of official Stata ado-files get installed in C:\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, **logit** is a built-in command, but the **logistic** command is defined by an ado-file named logistic.ado:

```
. which logit
built-in command:  logit

. which logistic
C:\STATA\ado\base\l\logistic.ado
*! version 3.1.9  01oct2002
```

This distinction makes no difference to most users, because **logit** and **logistic** work with similar ease and syntax when called.

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 that 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 just 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 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,

```
          display `iterate'
0
```

Thus, to increase the value of `iterate` by one, we write

```
          local iterate = `iterate' + 1
```

Global Macros

Global macros are similar to local macros, but once defined, they remain in memory and can be used by other programs. 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):

```
          global distance = 73
          display $distance * 2
146
```

Version

Stata's capabilities and features 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 6:

```
        version 6
```

Comments

Stata does not attempt to execute any line that begins with an asterisk. Such lines can therefore be used to insert comments and explanation 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 income education     // 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 income education     /// this part is the comment
      occupation age
```

will be executed as if we had typed

```
      summarize income education occupation age
```

With or without comments, the triple slash provides an easy way to include long command lines in a program. For example, the following lines would be read as one `table` command, even though they are separated by a hard return.

```
      table gender kids school if contam==1, contents(mean lived ///
         median lived count lived)
```

If our program has more than a few long commands, however, the `#delimit ;` approach (described earlier; also see **`help delimit`**) might be easier to write and read.

It is also possible to include comments in the middle of a command line, bracketed by `/*` and `*/`. For example,

```
      summarize income /* this is the comment */ education occupation
```

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.

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 8.0
   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. 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 nothing (on that line) follows the opening bracket, and that the closing bracket requires a line of its own.

The `foreach` command takes a different approach. Instead of specifying a set of consecutive numerical values, we give a list of items for which iteration occurs. These items could be variables, files, strings, or numerical 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 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 8.0
  local iterate = 1
  while `iterate' <= 6 {
     display `iterate'
  local iterate = `iterate' + 1
  }
end
```

A second example of a `while` loop appears in the *gossip.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.

```
* Perform simple regression and list observations with #
* largest absolute residuals.
*   listres1 Yvariable Xvariable # IDvariable
program listres1, sortpreserve
   version 8.0
   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 that the command `listresid` should be followed by four arguments. 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 (followed by a minus sign before the variable name) to sort the data in high-to-low order, with missing values last:

```
gsort -Absres
```

The option `sortpreserve` on the command line makes this program "sort-stable": it returns the data to their original order after the calculations are finished.

Dataset *nations.dta*, seen previously in Chapter 8, contains variables indicating life expectancy (*life*), per capita daily calories (*food*), and country name (*country*) for 109 countries. 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 `listres1`:

```
. do listres1.do
```

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 life food 5 country
```

```
     +--------------------------------------+
     | country    life       Yhat      Resid |
     |--------------------------------------|
  1. |   Libya      60    76.6901  -16.69011 |
  2. |  Bhutan      44   60.49577  -16.49577 |
  3. |  Panama      72   58.13118   13.86882 |
  4. |  Malawi      45   58.58232  -13.58232 |
  5. | Ecuador      66   52.45305   13.54695 |
     +--------------------------------------+
```

Life expectancies are lower than predicted in Libya, Bhutan, and Malawi. Conversely, life expectancies in Panama and Ecuador are higher than predicted, based on food supplies.

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
program listres2, sortpreserve
version 8.0
syntax varlist(min=1) [if] [in], Number(integer) [Id(string)]
   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 improvements, however, 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 life expectancy on *food* and *energy*, while restricting our analysis to only those countries where per capita GNP is above 500 dollars:

```
. do listres2.do

. listres2 life food energy if gnpcap > 500, n(6) i(country)

     +----------------------------------------+
     |  country   life        Yhat      Resid |
     |----------------------------------------|
  1. | YemenPDR     46    61.34964  -15.34964 |
  2. |  YemenAR     45    59.85839  -14.85839 |
  3. |    Libya     60    73.62516  -13.62516 |
  4. | S_Africa     55     67.9146   -12.9146 |
  5. | HongKong     76    64.64022   11.35978 |
     |----------------------------------------|
  6. |   Panama     72    61.77788   10.22212 |
     +----------------------------------------+
```

The `syntax` line in this example illustrates some general features of the command:

```
    syntax varlist(min=1) [if] [in], Number(integer) [Id(string)]
```

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(string)`, an actual command could be written:

```
. listres2 life food, number(6) id(country)
```

Or, equivalently,

```
. listres2 life food, n(6) i(country)
```

The contents of local macro `number` are required to be an integer, and `id` is a string (such as *country*, a variable's name).

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: Moving Autocorrelation

The preceding sections presented basic ideas and example short programs. In this section, we apply those ideas to a slightly longer program that defines a new statistical procedure. The procedure obtains moving autocorrelations through a time series, as proposed for ocean-atmosphere data by Topliss (2001). The following do-file, *gossip.do*, defines a program that makes available a new command called `gossip`. Comments, in lines that begin with `*` or in phrases set off by `//`, explain what the program is doing. Indentation of lines has no effect on the program's execution, but makes it easier for the programmer to read.

```
capture program drop gossip     // FOR WRITING & DEBUGGING; DELETE LATER
program gossip
version 8.0
* Syntax requires user to specify two variables (Yvar and TIMEvar), and
* the span of the moving window.  Optionally, the user can ask to generate
* a new variable holding autocorrelations, to draw a graph, or both.
syntax varlist(min=1 max=2 numeric), SPan(integer) [GENerate(string) GRaph]
if int(`span'/2) != (`span' - 1)/2 {
    display as error "Span must be an odd integer"
}
else {
* The first variable in `varlist' becomes Yvar, the second TIMEvar.
    tokenize `varlist'
        local Yvar `1'
        local TIMEvar `2'
    tempvar NEWVAR
    quietly gen `NEWVAR' = .
    local miss = 0
* spanlo and spanhi are local macros holding the observation number at the
* low and high ends of a particular window.  spanmid holds the observation
* number at the center of this window.
    local spanlo = 0
    local spanhi = `span'
    local spanmid = int(`span'/2)
    while `spanlo' <=  _N -`span' {
        local spanhi = `span' + `spanlo'
        local spanlo = `spanlo' + 1
        local spanmid = `spanmid' + 1
* The next lines check whether missing values exist within the window.
* If they do exist, then no autocorrelation is calculated and we
* move on to the next window.  Users are informed that this occurred.
        quietly summ `Yvar' in `spanlo'/`spanhi'
        if r(N) != `span' {
            local miss = 1
        }
* The value of NEWVAR in observation `spanmid' is set equal to the first
* row, first column (1,1) element of the row vector of autocorrelations
* r(AC) saved by corrgram.
        else {
            quietly corrgram `Yvar' in `spanlo'/`spanhi', lag(1)
            quietly replace `NEWVAR' = el(r(AC),1,1) in `spanmid'
        }
    }
    if "`graph'" != "" {
* The following graph command illustrates the use of comments to cause
* Stata to skip over line breaks, so it reads the next two lines as if
* they were one.
        graph twoway spike `NEWVAR' `TIMEvar', yline(0) ///
            ytitle("First-order autocorrelations of `Yvar' (span `span')")
    }
    if `miss' == 1 {
        display as error "Caution:  missing values exist"
    }
    if "`generate'" != "" {
        rename `NEWVAR' `generate'
        label variable `generate' ///
            "First-order autocorrelations of `Yvar' (span `span')"
    }
}
end
```

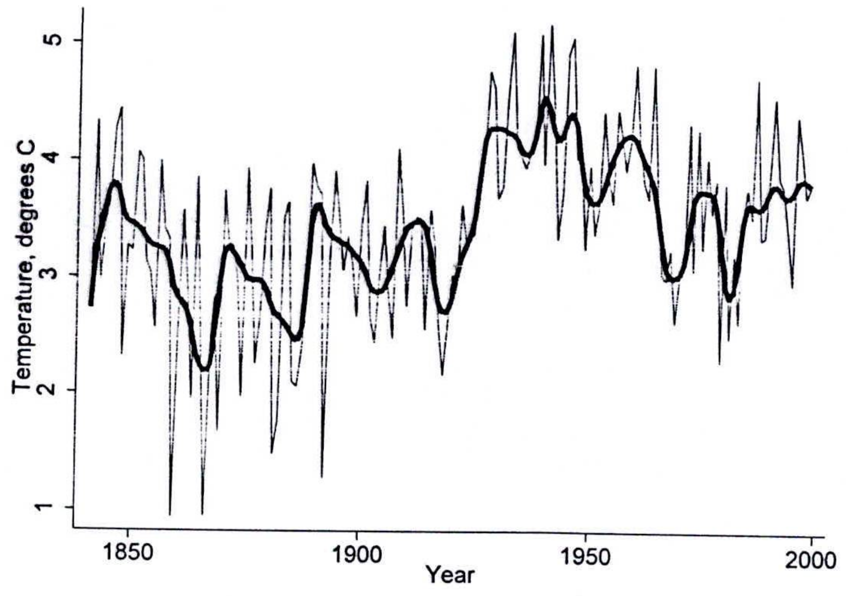
As the comments note, `gossip` requires time series (`tsset`) data. From an existing time series variable, `gossip` calculates a second time series consisting of lag-1 autocorrelation coefficients within a moving window of observations — for example, a moving 9-year span. Dataset *nao.dta* contains North Atlantic climate time series that can be used for illustration:

```
Contains data from C:\data\nao.dta
  obs:           159                          North Atlantic Oscillation &
                                                mean air temperature at
                                                Stykkisholmur, Iceland
 vars:             5                          1 Aug 2005 10:50
 size:         3,498 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
year            int    %ty                    Year
wNAO            float  %9.0g                  Winter NAO
wNAO4           float  %9.0g                  Winter NAO smoothed
temp            float  %9.0g                  Mean air temperature (C)
temp4           float  %9.0g                  Mean air temperature smoothed
-------------------------------------------------------------------------------
Sorted by:  year
```

The variable *temp* records annual mean air temperatures at Stykkishólmur in west Iceland from 1841 to 1999. *temp4* contains smoothed values of *temp* (see Chapter 13). Figure 14.1 graphs these two time series. To visually distinguish between raw (*temp*) and smoothed (*temp4*) variables, we connect the former with very thin lines, **clwidth(vthin)**, and the latter with thick lines, **clwidth(thick)**. Type **help linewidthstyle** for a list of other line-width choices.

```
. graph twoway line temp year, clpattern(solid) clwidth(vthin)
      || line temp4 year, clpattern(solid) clwidth(thick)
      || , ytitle("Temperature, degrees C") legend(off)
```

Figure 14.1



To calculate and graph a series of autocorrelations of *temp*, within a moving window of 9 years, we type the following commands. They produce the graph shown in Figure 14.2.

```
. do gossip.do

. gossip temp year, span(9) generate(autotemp) graph
```

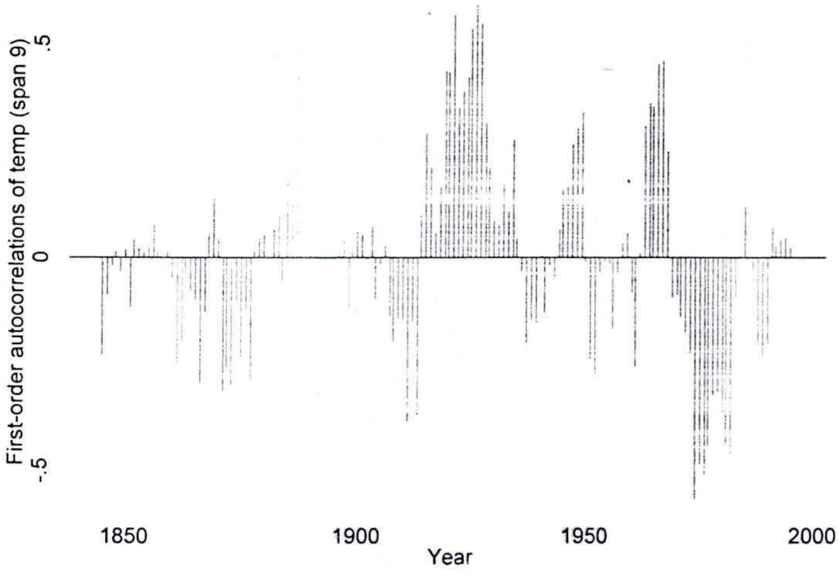


Figure 14.2

In addition to drawing Figure 14.2, `gossip` created a new variable named *autotemp*:

```
. describe autotemp

              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
autotemp        float  %9.0g                  First-order autocorrelations of
                                                temp (span 9)

. list year temp autotemp in 1/10

     +--------------------------+
     | year   temp    autotemp  |
     |--------------------------|
  1. | 1841   2.73           .  |
  2. | 1842   4.34           .  |
  3. | 1843   2.97           .  |
  4. | 1844   3.41           .  |
  5. | 1845   3.62   -.2324837  |
     |--------------------------|
  6. | 1846   4.28   -.0883512  |
  7. | 1847   4.45   -.0194607  |
  8. | 1848   2.32    .0175247  |
  9. | 1849   3.27     -.03303  |
 10. | 1850   3.23    .0181154  |
     +--------------------------+
```

autotemp values are missing for the first four years (1841 to 1844). In 1845, the *autotemp* value (–.2324837) equals the lag-1 autocorrelation of *temp* over the 9-year span from 1841 to 1849. This is the same coefficient we would obtain by typing the following command:

```
. corrgram temp in 1/9, lag(1)
                                                -1       0       1 -1       0       1
 LAG       AC        PAC        Q      Prob>Q  [Autocorrelation]  [Partial Autocor]
-------------------------------------------------------------------------------------
1       -0.2325  -0.2398    .66885  0.4135          -|                   -|
```

In 1846, *autotemp* (–.0883512) equals the lag-1 autocorrelation of *temp* over the 9 years from 1842 to 1850. and so on through the data. *autotemp* values are missing for the last four years in the data (1996 to 1999), as they are for the first four.

The pronounced Arctic warming of the 1920s, visible in the temperatures of Figure 14.1, manifests in Figure 14.2 as a period of consistently positive autocorrelations. A briefer period of positive autocorrelations in the 1960s coincides with a cooling climate. Topliss (2001) suggests interpretation of such autocorrelations as indicators of changing feedbacks in ocean-atmosphere systems.

The do-file *gossip.do* was written incrementally, starting with input components such as the syntax statement and span macros, running the do-file to check how these work, and then adding other components. Not all of the trial runs produced satisfactory results. Typing the following command causes Stata to display programs line-by-line as they execute, so we can see exactly where an error occurs:

```
. set trace on
```

Later, we can turn this feature off by typing

```
. set trace off
```

gossip.do contains a first line, `capture program drop gossip`, that discards the program from memory before defining it again. This is helpful during the writing and debugging stage, when a previous version of our program might have been incomplete or incorrect. Such lines should be deleted once the program is mature, however. The next section describes further steps toward making `gossip` available as a regular Stata command.

Ado-File

Once we believe our do-file defines a program that we will want to use again, we can create an ado-file to make it available like any other Stata command. For the previous example, *gossip.do*, the change involves two steps:

1. With the Do-file Editor, delete the initial "DELETE LATER" line that had been inserted to streamline the program writing and debugging phase. We can also delete the comment lines. Doing so removes useful information, but it makes the program more compact and easier to read.
2. Save our modified file, renaming it to have an .ado extension (for example, *gossip.ado*), in a new directory. The recommended location is in C:\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 `gossip` as a regular command within Stata. A listing of *gossip.ado* follows.

```
*!  version 2.0
*!  L. Hamilton, Statistics with Stata (2004)
program gossip
version 8.0
syntax varlist(min=1 max=2 numeric), SPan(integer) [GENerate(string) GRaph]
if int(`span'/2) != (`span' - 1)/2 {
    display as error "Span must be an odd integer"
}
else {
    tokenize `varlist'
        local Yvar `1'
        local TIMEvar `2'
    tempvar NEWVAR
    quietly gen `NEWVAR' = .
    local miss = 0
    local spanlo = 0
    local spanhi = `span'
    local spanmid = int(`span'/2)
    while `spanlo' <= _N -`span' {
        local spanhi = `span' + `spanlo'
        local spanlo = `spanlo' + 1
        local spanmid = `spanmid' + 1
        quietly summ `Yvar' in `spanlo'/`spanhi'
        if r(N) != `span' {
            local miss = 1
        }
        else {
            quietly corrgram `Yvar' in `spanlo'/`spanhi', lag(1)
            quietly replace `NEWVAR' = el(r(AC),1,1) in `spanmid'
        }
    }
    if "`graph'" != "" {
        graph twoway spike `NEWVAR' `TIMEvar', yline(0) ///
            ytitle("First-order autocorrelations of `Yvar' (span `span')")
    }
    if `miss' == 1 {
        display as error "Caution:  missing values exist"
    }
    if "`generate'" != "" {
        rename `NEWVAR' `generate'
        label variable `generate' ///
            "First-order autocorrelations of `Yvar' (span `span')"
    }
}
end
```

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 *gossip.ado*, not Stata (an earlier version of *gossip.ado* appeared in a previous edition of this book). The Stata version suitable for this program is specified as 8.0 by the `version` command a few lines later. Although the *! comments do not affect how the program runs, they are visible to a **`which`** command:

```
. which gossip
c:\ado\personal\gossip.ado
*!  version 2.0
*!  L. Hamilton, Statistics with Stata (2004)
```

Once *gossip.ado* has been saved in the C:\ado\personal directory, the command `gossip` could be used at any time. If we are following the steps in this chapter, which previously defined a preliminary version of `gossip`, then before running the new ado-file version we should drop the old definition from memory by typing

```
. program drop gossip
```

We are now prepared to run the final, ado-file version. To see a graph of span-15 autocorrelations of variable *wNAO* from dataset *nao.dta*, for example, we would simply open *nao.dta* and type

```
. gossip wNAO year, span(15) graph
```

Help File

Help files are an integral aspect of using Stata. For a user-written program such as *gossip.ado*, they become even more important because no documentation exists in the printed manuals. We can write a help file for *gossip.ado* by using Stata's Do-file Editor to create a text file named *gossip.hlp*. This help file should be saved in the same ado-file directory (for example, C:\ado personal) as *gossip.ado*.

Any text file, saved in one of Stata's recognized ado-file directories with a name of the form *filename.hlp*, 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 *gossip1.hlp*. Typing **`help gossip1`** at any time would then cause Stata to display the text.

```
help for gossip                    L. Hamilton

Moving first-order autocorrelations

gossip yvar timevar, span(#) [ generate(newvar) graph ]

Description

calculates first-order autocorrelations of time series
yvar, within a moving window of span #.  For example, if we
specify span(7) gen(new), then the first
through 3rd values of new are missing.  The 4th value of new
equals the lag-1 autocorrelation of yvar across observations 1
through 7.  The 5th value of new equals the lag-1 autocorrelation
of yvar across observations 2 through 8, and so forth.  The last
3 values of new are missing.  See Topliss (2001) for a rationale
and applications of this statistic to atmosphere-ocean data.
Statistics with Stata (2004) discusses the gossip program itself.

gossip requires tsset data.  timevar is the time
variable to be used for graphing.

Options

span(#)    specifies the width of the window for
       calculating autocorrelations.  This option is required;
       # should be an odd integer.

gen(newvar)   creates a new variable holding the
       autocorrelation coefficients.
```

```
graph      requests a spike plot of lag-1 autocorrelations vs.
      timevar.

Examples

     . gossip water month, span(13) graph
     . gossip water month, span(9) gen(autowater)
     . gossip water month, span(17) gen(autowater) graph

References

Hamilton, Lawrence C.  2004.  Statistics with Stata.  Pacific Grove,
 CA:  Duxbury.

Topliss, Brenda J.  2001.  "Climate variability I:  A conceptual approach to
ocean-atmosphere feedback."  In Abstracts for AGU Chapman Conference, The
North Atlantic Oscillation, Nov. 28 - Dec 1, 2000, Ourense, Spain.
```

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. The following is an SMCL version of the help file for `gossip`. Once this file has been saved in C:\ado\personal with the file name *gossip.hlp*, typing **`help gossip`** will produce a readable and official-looking display.

```
{smcl}
{* 1aug2003}{...}
{hline}
help for {hi:gossip}{right:(L. Hamilton)}
{hline}

{title:Moving first-order autocorrelations}

{p 8 12}{cmd:gossip} {it:yvar timevar} {cmd:,} {cmdab:sp:an}{cmd:(}
{it:#}{cmd:)} [ {cmdab:gen:erate}{cmd:(}{it:newvar}{cmd:)}
{cmdab:gr:aph} ]


{title:Description}

{p}{cmd:gossip} calculates first-order autocorrelations of time series
{it:yvar}, within a moving window of span {it:#}.  For example, if we
specify {cmd:span(}7{cmd:)} {cmd:gen(}{it:new}{cmd:)}, then the first
through 3rd values of {it:new} are missing.  The 4th value of {it:new}
equals the lag-1 autocorrelation of {it:yvar} across observations 1
through 7.  The 5th value of {it:new} equals the lag-1 autocorrelation
of {it:yvar} across observations 2 through 8, and so forth.  The last
3 values of {it:new} are missing.  See Topliss (2001) for a rationale
and applications of this statistic to atmosphere-ocean data.
{browse "http://www.stata.com/bookstore/sws.html":Statistics with Stata}
 (2004) discusses the {cmd:gossip} program itself.{p_end}

{p}{cmd:gossip} requires {cmd:tsset} data.  {it:timevar} is the time
variable to be used for graphing.{p_end}


{title:Options}

{p 0 4}{cmd:span(}{it:#}{cmd:)} specifies the width of the window for
calculating autocorrelations.  This option is required; {it:#} should be
 an odd integer.
```

```
{p 0 4}{cmd:gen(}{it:newvar}{cmd:)} creates a new variable holding the
autocorrelation coefficients.

{p 0 4}{cmd:graph} requests a spike plot of lag-1 autocorrelations vs.
{it:timevar}.


{title:Examples}

{p 8 12}{inp:. gossip water month, span(13) graph}{p_end}
{p 8 12}{inp:. gossip water month, span(9) gen(autowater)}{p_end}
{p 8 12}{inp:. gossip water month, span(17) gen(autowater) graph}{p_end}


{title:References}

{p 0 4}Hamilton, Lawrence C.  2004.
{browse "http://www.stata.com/bookstore/sws.html":Statistics with Stata}.
 Pacific Grove, CA:  Duxbury.{p_end}

{p 0 4}Topliss, Brenda J.  2001.  "Climate variability I:  A conceptual
approach to ocean-atmosphere feedback."  In Abstracts for AGU Chapman
Conference, The North Atlantic Oscillation, Nov. 28 - Dec 1, 2000, Ourense,
Spain. citation.{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.

`{hline}`	Draw a horizontal line.
`{hi:gossip}`	Highlight the text "gossip".
`{title:Moving...}`	Display the text "Moving . . ." as a title.
`{right:L Hamilton}`	Right-justify the text "L. Hamilton".
`{p 8 12}`	Format the following text as a paragraph, with the first line indented 8 columns and subsequent lines indented 12.
`{cmd:gossip}`	Display the text "gossip" as a command. That is, show "gossip" with whatever colors and font attributes are presently defined as appropriate for a command.
`{it:yvar}`	Display the text "yvar" in italics.
`{cmdab:sp:an}`	Display "span" as a command, with the letters "sp" marked as the minimum abbreviation.
`{p}`	Format the following text as a paragraph, until terminated by `{p_end}`.
`{browse "http://www.stata.com/bookstore/sws.html":Statistics...}`	Link the text "Statistics with Stata" to the web address (URL) http://www.stata.com/bookstore/sws.html. Clicking on the words "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.

Matrix Algebra

Matrix algebra provides essential tools for statistical modeling. Stata's matrix commands and matrix programming language (Mata) are too diverse to describe adequately here; the subject requires its own reference manual (*Mata Reference Manual*), in addition to many pages in the *Programming Reference Manual* and *User's Guide.* Consult these sources for information about the Mata language, which is new with Stata 9. The examples in this section illustrate earlier matrix commands, which also still work (hence the placement of **`version 8.0`** commands at the start of each program).

The built-in Stata command **`regress`** performs ordinary least squares (OLS) regression, among other things. But as an exercise, we could write an OLS program ourselves. *ols1.do* (following) defines a primitive regression program that does nothing except calculate and display the vector of estimated regression coefficients according to the familiar OLS equation:

$$\mathbf{b} = (\mathbf{X'X})^{-1}\mathbf{X'y}$$

```
* A very simple program, "ols1" estimates linear regression
* coefficients using ordinary least squares (OLS).
program ols1
   version 8.0
* The syntax allows only for a variable list with one or more
* numeric variables.
   syntax varlist(min=1 numeric)
* "tempname..." assigns names to temporary matrices to be used in this
* program.  When ols1 has finished, these matrices will be dropped.
   tempname crossYX crossX crossY b
* "matrix accum..." forms a cross-product matrix.  The K variables in
* varlist, and the N observations with nonmissing values on all K variables,
* comprise an N row, K column data matrix we might call yX.
* The cross product matrix crossYX equals the transpose of yX times yX.
* Written algebraically:
*          crossYX = (yX)'yX
   quietly matrix accum `crossYX' = `varlist'
* Matrix crossX extracts rows 2 through K, and columns 2 through K,
* from crossYX:
*          crossX = X'X
   matrix `crossX' = `crossYX'[2...,2...]
* Column vector crossY extracts rows 2 through K, and column 1 from crossYX:
*          crossY = X'y
   matrix `crossY' = `crossYX'[2...,1]
* The column vector b contains OLS regression coefficients, obtained by
* the classic estimating equation:
*          b = inverse(X'X)X'y
   matrix `b' = syminv(`crossX') * `crossY'
* Finally, we list the coefficient estimates, which are the contents of b.
   matrix list `b'
end
```

Comments explain each command in *ols1.do*. A comment-free version named *ols2.do* (following) gives a clearer view of the matrix commands:

```
program ols2
    version 8.0
    syntax varlist(min=1 numeric)
    tempname crossYX crossX crossY b
    quietly matrix accum `crossYX' = `varlist'
    matrix `crossX' = `crossYX'[2...,2...]
    matrix `crossY' = `crossYX'[2...,1]
    matrix `b' = syminv(`crossX') * `crossY'
    matrix list `b'
end
```

Neither *ols1.do* nor *ols2.do* make any provision for `in` or `if` qualifiers, syntax errors, or options. They also do not calculate standard errors, confidence intervals, or the other ancillary statistics we usually want with regression. To see just what they do accomplish, we will analyze a small dataset on nuclear power plants (*reactor.dta*):

```
Contains data from c:\data\reactor.dta
  obs:            5                          Reactor decommissioning costs
                                               (from Brown et al. 1986)
 vars:            6                          1 Aug 2005 10:50
 size:          130 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
site            str14  %14s                   Reactor site
decom           byte   %9.0g                  Decommissioning cost, millions
capacity        int    %8.0g                  Generating capacity, megawatts
years           byte   %9.0g                  Years in operation
start           int    %8.0g                  Year operations started
close           int    %8.0g                  Year operations closed
-------------------------------------------------------------------------------
Sorted by:  start
```

The cost of decommissioning a reactor increases with its generating capacity and with the number of years in operation, as can be seen by using **`regress`**:

```
. regress decom capacity years

      Source |       SS       df       MS              Number of obs =       5
-------------+------------------------------           F(  2,     2) =  189.42
       Model |  4666.16571     2  2333.08286           Prob > F      =  0.0053
    Residual |  24.6342883     2  12.3171442           R-squared     =  0.9947
-------------+------------------------------           Adj R-squared =  0.9895
       Total |     4690.80     4     1172.70           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
------------------------------------------------------------------------------
```

Our home-brewed program *ols2.do* yields exactly the same regression coefficients:

```
. do ols2.do

. ols2 decom capacity years

__000003[3,1]
                 decom
capacity     .1758739
   years    3.8993139
   _cons   -11.399633
```

Although its results are correct, the minimalist `ols2` program lacks many features we would want in a useful modeling command. The following ado-file, *ols3.ado*, defines an improved program named `ols3` . This program permits `in` and `if` qualifiers, and optionally allows specification of the level for confidence intervals. It calculates and neatly displays regression coefficients in a table with their standard errors, *t* tests, and confidence intervals.

```
*! version 2.0 1aug2003
*! Matrix demonstration:  more complete OLS regression program.
program ols3, eclass
    version 8.0
    syntax varlist(min=1 numeric) [in] [if] [, Level(integer $S_level)]
    marksample touse
    tokenize "`varlist'"
    tempname crossYX crossX crossY b hat V
    quietly matrix accum `crossYX' = `varlist' if `touse'
    local nobs = r(N)
    local df = `nobs' - (rowsof(`crossYX') - 1)
    matrix `crossX' = `crossYX'[2...,2...]
    matrix `crossY' = `crossYX'[2...,1]
    matrix `b' = (syminv(`crossX') * `crossY')'
    matrix `hat' = `b' * `crossY'
    matrix `V' = syminv(`crossX') * (`crossYX'[1,1] - `hat'[1,1])/`df'
    ereturn post `b' `V', dof(`df') obs(`nobs') depname(`1') ///
        esample(`touse')
    ereturn local depvar "`1'"
    ereturn local cmd "ols3"
    if `level' < 10 | `level' > 99 {
        display as error "level( ) must be between 10 and 99 inclusive."
        exit 198
    }
    ereturn display, level(`level')
end
```

Because *ols3.ado* is an ado-file, we can simply type `ols3` as a command:

```
. ols3 decom capacity years

------------------------------------------------------------------------------
       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
------------------------------------------------------------------------------
```

ols3.ado contains familiar elements including `syntax` and `marksample` commands, as well as `matrix` operations built upon those seen earlier in *ols1.do* and *ols2.do*. Note the use of a right single quote (') as the "matrix transpose" operator. We write the transpose of the coefficients vector `(syminv(`crossX') * `crossY')` as follows:

```
(syminv(`crossX') * `crossY')'
```

The `ols3` program is defined as e-class, indicating that this is a statistical model-estimation command:

```
program ols3, eclass
```

E-class programs store their results with `e()` designations. After the previous **`ols3`** command, these have the following contents:

```
. ereturn list

scalars:
                  e(N) =  5
               e(df_r) =  2

macros:
                e(cmd) : "ols3"
             e(depvar) : "decom"

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

functions:
             e(sample)

. display e(N)
5

. matrix list e(b)

e(b)[1,3]
      capacity        years        _cons
y1    .1758739    3.8993139   -11.399633

. matrix list e(V)

symmetric e(V)[3,3]
             capacity        years        _cons
capacity    .00061392
   years   -.00216732     .0698591
   _cons   -.01492755     -.942626    18.751591
```

The `e()` results from e-class programs remain in memory until the next e-class command. In contrast, r-class programs such as **`summarize`** store their results with `r()` designations, and these remain in memory only until the next e- or r-class command.

Several `ereturn` lines in *ols3.ado* save the `e()` results, then use these in the output display :

```
ereturn post `b' `V', dof(`df') obs(`nobs') depname(`1') ///
     esample(`touse')
```

The above command sets the contents of `e()` results, including the coefficient vector (b) and the variance–covariance matrix (V). This makes all the post-estimation features detailed in **`help estimates`** and **`help postest`** available. Options specify the residual degrees of freedom (`df`), number of observations used in estimation (`nobs`), dependent variable name (`1' . meaning the contents of the first macro obtained when we `tokenize varlist` ), and estimation sample marker ( `touse`).

`ereturn local depvar "`1'"`

This command sets the name of the dependent variable, macro `1` after `tokenize varlist`, to be the contents of macro `e(depvar)`.

`ereturn local cmd "ols3"`

This sets the name of the command, `ols3`, as the contents of macro `e(cmd)`.

`ereturn display, level(`level')`

The `ereturn display` command displays the coefficient table based on our previous `ereturn post`. This table follows a standard Stata format: its first two columns contain coefficient estimates (from `b`) and their standard errors (square roots of diagonal elements from `V`). Further columns are *t* statistics (first column divided by second), two-tail *t* probabilities, and confidence intervals based on the level specified in the `ols3` command line (or defaulting to 95%).

Bootstrapping

Bootstrapping refers to a process of repeatedly drawing random samples, with replacement, from the data at hand. Instead of trusting theory to describe the sampling distribution of an estimator, we approximate that distribution empirically. Drawing *k* bootstrap samples of size *n* (from an original sample also size *n)* yields *k* new estimates. The distribution of these bootstrap estimates provides an empirical basis for estimating standard errors or confidence intervals (Efron and Tibshirani 1986; for an introduction, see Stine in Fox and Long 1990). Bootstrapping seems most attractive in situations where the statistic of interest is theoretically intractable, or where the usual theory regarding that statistic rests on untenable assumptions.

Unlike Monte Carlo simulations, which fabricate their data, bootstrapping typically works from real data. For illustration, we turn to *islands.dta*, containing area and biodiversity measures for eight Pacific Island groups (from Cox and Moore 1993).

```
Contains data from c:\data\islands.dta
  obs:             8                          Pacific island biodiversity
                                                (Cox & Moore 1993)
 vars:             4                          1 Aug 2005 10:50
 size:           208 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
island          str15  %15s                   Island group
area            float  %9.0g                  Land area, km^2
birds           byte   %8.0g                  Number of bird genera
plants          int    %8.0g                  Number flowering plant genera
-------------------------------------------------------------------------------
Sorted by:
```

Suppose we wish to form a confidence interval for the mean number of bird genera. The usual confidence interval for a mean derives from a normality assumption. We might hesitate to make this assumption, however, given the skewed distribution that, even in this tiny sample ($n = 8$), almost leads us to reject normality:

```
. sktest birds

                   Skewness/Kurtosis tests for Normality
                                                   ------- joint ------
    Variable |  Pr(Skewness)   Pr(Kurtosis)  adj chi2(2)    Prob>chi2
-------------+-------------------------------------------------------
       birds |       0.079          0.181           4.75        0.0928
```

Bootstrapping provides a more empirical approach to forming confidence intervals. An r-class command, **`summarize, detail`** unobtrusively stores its results as a series of macros. Some of these macros are:

`r(N)`	Number of observations
`r(mean)`	Mean
`r(skewness)`	Skewness
`r(min)`	Minimum
`r(max)`	Maximum
`r(p50)`	50th percentile or median
`r(Var)`	Variance
`r(sum)`	Sum
`r(sd)`	Standard deviation

Stored results simplify the job of bootstrapping any statistic. To obtain bootstrap confidence intervals for the mean of *birds*, based on 1,000 resamplings, and save the results in new file *boot1.dta*, type the following command. The output includes a note warning about the potential problem of missing values, but that does not apply to these data.

```
. bs "summarize birds, detail"  "r(mean)", rep(1000) saving(boot1)

command:      summarize birds , detail
statistic:    _bs_1       = r(mean)

Warning:  Since summarize is not an estimation command or does not set
          e(sample), bootstrap has no way to determine which observations are
          used in calculating the statistics and so assumes that all
          observations are used.  This means no observations will be excluded
          from the resampling due to missing values or other reasons.

          If the assumption is not true, press Break, save the data, and drop
          the observations that are to be excluded.  Be sure the dataset in
          memory contains only the relevant data.

Bootstrap statistics                              Number of obs    =         8
                                                  Replications     =      1000

------------------------------------------------------------------------------
Variable     |  Reps  Observed       Bias  Std. Err.  [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _bs_1 |  1000    47.625   -.475875  12.39088    23.30986   71.94014   (N)
             |                                            25.75    74.8125   (P)
             |                                               27      78.25  (BC)
------------------------------------------------------------------------------
Note:  N   = normal
       P   = percentile
       BC  = bias-corrected
```

The **bs** command states in double quotes what analysis is to be bootstrapped (**"summ *birds*, detail"**). Following this comes the statistic to be bootstrapped, likewise in its own double quotes (**"r(mean) "**). More than one statistic could be listed, each separated by a space. The example above specifies two options:

`rep(1000)`	Calls for 1,000 repetitions, or drawing 1,000 bootstrap samples.
`saving(boot1)`	Saves the 1,000 bootstrap means in a new dataset named *boot1.dta*.

The **bs** results table shows the number of repetitions performed and the "observed" (original-sample) value of the statistic being bootstrapped — in this case, the mean *birds* value 47.625. The table also shows estimates of bias, standard error, and three types of confidence intervals. "Bias" here refers to the mean of the *k* bootstrap values of our statistic (for example, the mean of the 1,000 bootstrap means of *birds*), minus the observed statistic. The estimated standard error equals the standard deviation of the *k* bootstrap statistic values (for example, the standard deviation of the 1,000 bootstrap means of *birds*). This bootstrap standard error (12.39) is less than the conventional standard error (13.38) calculated by **ci** :

```
. ci birds

    Variable |        Obs        Mean    Std. Err.       [95% Conf. Interval]
-------------+---------------------------------------------------------------
       birds |          8      47.625    13.38034        15.98552    79.26448
```

Normal-approximation (N) confidence intervals in the **bs** table are obtained as follows:

observed sample statistic $\pm t \times$ bootstrap standard error

where t is chosen from the theoretical t distribution with $k - 1$ degrees of freedom. Their use is recommended when the bootstrap distribution appears unbiased and approximately normal.

Percentile (P) confidence intervals simply use percentiles of the bootstrap distribution (for a 95% interval, the 2.5th and 97.5th percentiles) as lower and upper bounds. These might be appropriate when the bootstrap distribution appears unbiased but nonnormal.

The bias-corrected (BC) interval also employs percentiles of the bootstrap distribution, but chooses these percentiles following a normal-theory adjustment for the proportion of bootstrap values less than or equal to the observed statistic. When substantial bias exists (by one guideline, when bias exceeds 25% of one standard error), these intervals might be preferred.

Since we saved the bootstrap results in a file named *boot1.dta*, we can retrieve this and examine the bootstrap distribution more closely if desired. The **saving(*boot1*)** option created a dataset with 1,000 observations and a variable named *_bs_1*, holding the mean of each bootstrap sample.

```
Contains data from c:\data\boot1.dta
  obs:         1,000                          bs: summarize birds, detail
 vars:             1                          1 Aug 2005 15:10
 size:         8,000 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
_bs_1           float  %9.0g                  r(mean)
-------------------------------------------------------------------------------
Sorted by:
```

```
. summarize

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
       _bs_1 |      1000    47.14912    12.39088      14.625       92.5
```

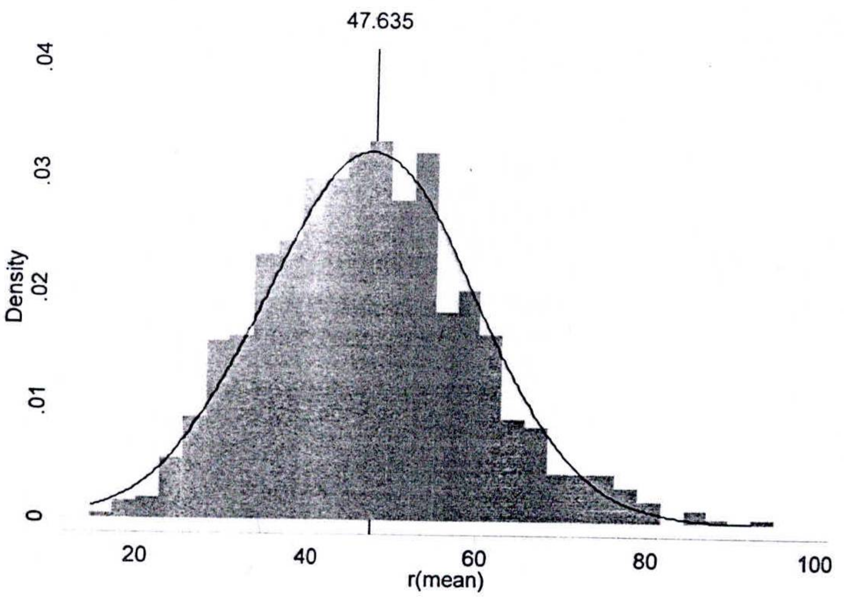
Note that the standard deviation of these 1,000 bootstrap means equals the standard error (12.82) shown earlier in the **bs** results table. The mean of the 1,000 means minus the observed (original sample) mean equals the bias:

$$47.14912 - 47.625 = -.47588$$

Figure 14.3 shows the distribution of these 1,000 sample means, with the original-sample mean (47.625) marked by a vertical line. The distribution exhibits mild positive skew, but is not far from a theoretical normal curve.

```
. histogram _bs_1, norm bcolor(gs10) xaxis(1 2) xline(47.625)
      xlabel(47.635, axis(2)) xtitle("", axis(2))
```

Figure 14.3



Biologists have observed that biodiversity, or the number of different kinds of plants and animals, tends to increase with island size. In *islands.dta*, we have data to test this proposition with respect to birds and flowering plants. As expected, a strong linear relationship exists between *birds* and *area*:

```
. regress birds area

      Source |       SS       df       MS              Number of obs =       8
-------------+------------------------------           F(  1,     6) =  162.96
       Model |  9669.83255     1  9669.83255           Prob > F      =  0.0000
    Residual |  356.042449     6  59.3404082           R-squared     =  0.9645
-------------+------------------------------           Adj R-squared =  0.9586
       Total |   10025.875     7  1432.26786           Root MSE      =  7.7033

------------------------------------------------------------------------------
       birds |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        area |   .0026512   .0002077    12.77   0.000      .002143    .0031594
       _cons |   13.97169    3.79046     3.69   0.010     4.696773    23.24662
------------------------------------------------------------------------------
```

An e-class command, **regress** saves a set of `e()` results as noted earlier in this chapter. It also creates or updates a set of system variables containing the model coefficients (`_b[`*varname*`]`) and standard errors (`_se[`*varname*`]`). To bootstrap the slope and *y* intercept from the previous regression, saving the results in file *boot2.dta*, type

```
. bs "regress birds area" "_b[area] _b[_cons]", rep(1000)
      saving(boot2)

command:      regress birds area
statistics:   _bs_1      = _b[area]
              _bs_2      = _b[_cons]

Bootstrap statistics                             Number of obs    =         8
                                                 Replications     =      1000

------------------------------------------------------------------------------
Variable     | Reps  Observed       Bias  Std. Err. [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _bs_1 | 1000  .0026512 -.0000737  .0003345   .0019947   .0033077   (N)
             |                                      .0019759   .0029066   (P)
             |                                        .00199   .0029246  (BC)
       _bs_2 | 1000  13.97169  .6230986  3.637705   6.833275   21.11011   (N)
             |                                      7.891942   21.74494   (P)
             |                                      6.949539   19.73012  (BC)
------------------------------------------------------------------------------
Note:  N   = normal
       P   = percentile
       BC  = bias-corrected
```

The bootstrap distribution of coefficients on *area* is severely skewed (skewness = 4.12). Whereas the bootstrap distribution of means (Figure 14.3) appeared approximately normal, and produced bootstrap confidence intervals narrower than the theoretical confidence interval, in this regression example bootstrapping obtains larger standard errors and wider confidence intervals.

In a regression context, **bs** ordinarily performs what is called "data resampling," or resampling intact observations. An alternative procedure called "residual resampling" (resampling only the residuals) requires a bit more programming work. Two additional commands make such do-it-yourself bootstrapping easier:

bsample	Draws a sample with replacement from the existing data, replacing the data in memory.

bootstrap Runs a user-defined program `reps()` times on bootstrap samples of size `size()`.

The *Base Reference Manual* gives examples of programs for use with **bootstrap**.

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. Below we see a file defining an r-class program (one capable of storing `r()` results) named `central`. 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, `central` 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 central, rclass
    version 8.0
    drop _all
    set obs 100
    generate x = invnorm(uniform())
    summarize x, detail
    return scalar xmean = r(mean)
    return scalar xmedian = r(p50)
    generate w = invnorm(uniform())
    replace w = 10*w if uniform() < .05
    summarize w, detail
    return scalar wmean = r(mean)
    return scalar wmedian = r(p50)
end
```

Because we defined **central** as an r-class command, like **summarize**, it can store its results in `r()` macros. **central** creates four such macros: `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 **central** 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 "central"  xmean = r(xmean)  xmedian = r(xmedian)
      wmean = r(wmean)  wmedian = r(wmedian), reps(5000)

command:       central
statistics:    xmean       = r(xmean)
               xmedian     = r(xmedian)
               wmean       = r(wmean)
               wmedian     = r(wmedian)
```

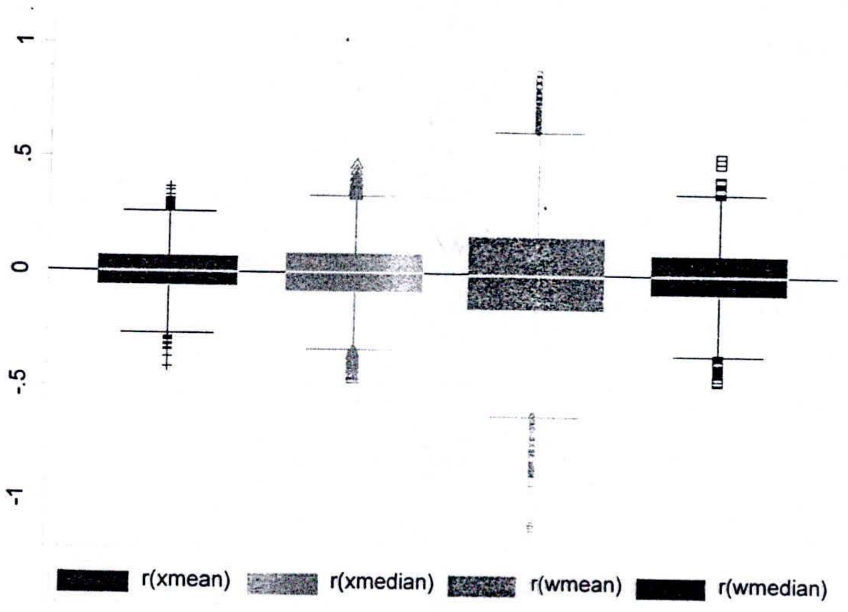
This command creates new variables *xmean*, *xmedian*, *wmean*, and *wmedian*, based on the `r()` results from each iteration of **central**.

```
. describe

Contains data
  obs:          5,000                          simulate: central
 vars:              4                          1 Aug 2005 17:50
 size:        100,000 (99.6% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      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   -.0015915    .0987788  -.4112561   .3699467
     xmedian |      5000   -.0015566    .1246915  -.4647848   .4740642
       wmean |      5000   -.0004433    .2470823   -1.11406   .8774976
     wmedian |      5000    .0030762    .1303756  -.4584521   .5152998
```

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. The standard deviation of *xmedian* is .125, noticeably larger than the standard deviation of *xmean* (.099). When applied to the 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* (.130 vs. .247). 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 14.4 draws the comparison graphically, with box plots (and, incidentally, demonstrates how to control the shapes of box plot outlier-marker symbols).

```
. graph box xmean xmedian wmean wmedian, yline(0) legend(col(4))
      marker(1, msymbol(+)) marker(2, msymbol(Th))
      marker(3, msymbol(Oh)) marker(4, msymbol(Sh))
```

Figure 14.4



Our final 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. *y1* is a linear function of *x* plus standard normal errors. *y2* is also a linear function of *x*, but adding contaminated normal errors. These variables permit us to explore how various regression methods behave in the presence of normal and nonnormal errors. 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). Differences among these methods were discussed in Chapter 9. Program `regsim` applies each method to the regression of *y1* on *x* and then to the regression of *y2* on *x*. For this exercise, the program is defined by an ado-file, *regsim.ado*, saved in the C:\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 8.0
    if "`1'" == "?" {
        #delimit ;
        global S_1 "b1 b1r se1r b1q se1q se1qb
            b2 b2r se2r b2q se2q se2qb";
        #delimit cr
        exit
    }
    drop _all
    set obs 100
    generate x = invnorm(uniform())
    generate e = invnorm(uniform())
    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 uniform() < .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 have 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 5 to 10 iterations for **`rreg`**, about 5 for **`qreg`**, and several thousand for **`bsqreg`** with its 500 bootstrap re-estimations of about 5 iterations each, *per sample*. Thus, a single execution of *regsim* demands more than two thousand regressions. The following command calls for five repetitions.

```
. simulate "regsim"  b1 = r(B1)  b1r = r(B1R)  se1r = r(SE1R)
      b1q = r(B1Q)  se1q = r(SE1Q)  se1qb = r(SE1QB)  b2 = r(B2)
      b2r = r(B2R)  se2r = r(SE2R)  b2q = r(B2Q)  se2q = r(SE2Q)
      se2qb = r(SE2QB), reps(5)
```

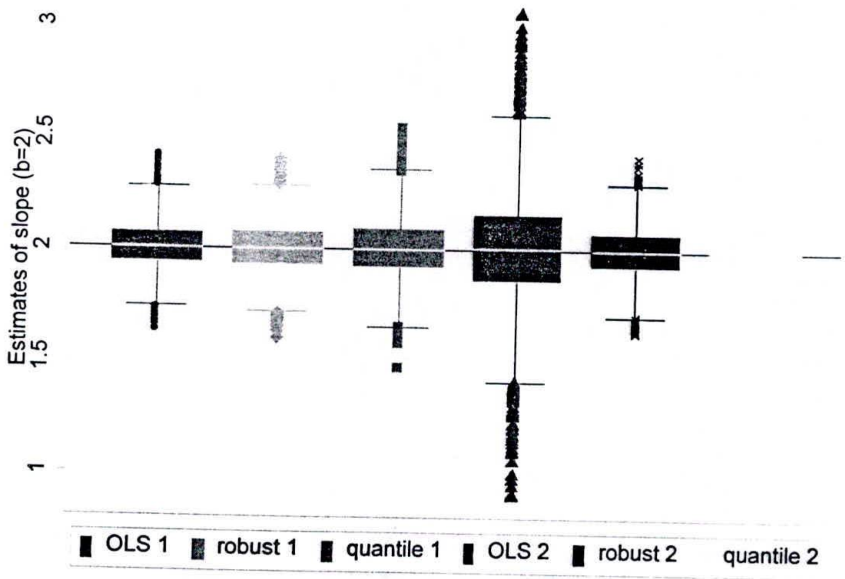
You might want to run a small simulation like this as a trial to get a sense of the time required on your computer. For research purposes, however, we would need a much larger experiment. Dataset *regsim.dta* contains results from an overnight experiment involving 5,000 repetitions of `regsim`— more than 10 million regressions. The regression coefficients and standard error estimates produced by this experiment are summarized below.

```
. describe

Contains data from C:\data\regsim.dta
  obs:         5,000                          Monte Carlo estimates of b in
                                                5000 samples of n=100
 vars:            12                          2 Aug 2005 18:17
 size:       260,000 (99.0% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
b1              float  %9.0g                  OLS b (normal errors)
b1r             float  %9.0g                  Robust b (normal errors)
se1r            float  %9.0g                  Robust SE[b] (normal errors)
b1q             float  %9.0g                  Quantile b (normal errors)
se1q            float  %9.0g                  Quantile SE[b] (normal errors)
se1qb           float  %9.0g                  Quantile bootstrap SE[b]
                                                (normal errors)
b2              float  %9.0g                  OLS b (contaminated errors)
b2r             float  %9.0g                  Robust b (contaminated errors)
se2r            float  %9.0g                  Robust SE[b] (contaminated
                                                errors)
b2q             float  %9.0g                  Quantile b (contaminated errors)
se2q            float  %9.0g                  Quantile SE[b] (contaminated
                                                errors)
se2qb           float  %9.0g                  Quantile bootstrap SE[b]
                                                (contaminated errors)
-------------------------------------------------------------------------------
Sorted by:

. summarize

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          b1 |      5000    2.000828     .102018   1.631245   2.404814
         b1r |      5000    2.000989    .1052277   1.603116   2.391946
        se1r |      5000    .1041399    .0109429   .0693786   .1515421
         b1q |      5000    2.001135    .1309186   1.471802   2.536621
        se1q |      5000    .1262578    .0281738   .0532731   .2371508
-------------+--------------------------------------------------------
       se1qb |      5000    .1362755     .032673   .0510808     .29979
          b2 |      5000    2.006001    .2484688   .9001114   3.050552
         b2r |      5000    2.000399    .1092553   1.633241   2.411423
        se2r |      5000    .1081348    .0119274   .0743103   .1560973
         b2q |      5000    2.000701     .137111   1.471802   2.536621
-------------+--------------------------------------------------------
        se2q |      5000    .1328431    .0299644   .0542015   .2594844
       se2qb |      5000    .1436366    .0346679   .0589409   .3006417
```

Figure 14.5 draws the distributions of coefficients as box plots. To make the plot more readable we use the **legend(symxsize(2) colgap(4))** options, which set the width of symbols and the gaps between columns within the legend at less than 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(2) colgap(4)
         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 14.5



All three regression methods (OLS, robust, and quantile) produced mean coefficient estimates for both models that are not significantly different from the true value, β = 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 |    5000    2.000399    .0015451    .1092553     1.99737    2.003428
------------------------------------------------------------------------------
Degrees of freedom: 4999

                           Ho: mean(b2r) = 2

     Ha: mean < 2             Ha: mean != 2               Ha: mean > 2
       t =   0.2585              t =   0.2585               t =   0.2585
   P < t =   0.6020          P > |t| =   0.7960           P > t =   0.3980
```

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 famous Gauss–Markov theorem would lead us to expect. The observed standard deviation of OLS coefficients is .1016, compared with .1047 for robust regression and .1282 for quantile regression. Relative efficiency, expressing the OLS coefficient's observed variance as a percentage of another estimator's observed variance, provides a standard way to compare such statistics:

```
. quietly summarize b1

. global Varb1 = r(Var)

. quietly summarize b1r

. display 100*($Varb1/r(Var))
93.992612

. quietly summarize b1q

. display 100*($Varb1/r(Var))
60.722696
```

The calculations above use the `r(Var)` variance result from **`summarize`**. We first obtain the variance of the OLS estimates *b1*, and place this into global macro `Varb1`. Next the variances of the robust estimates *b1r*, and the quantile estimates *b1q*, are obtained and each 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 61% 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:

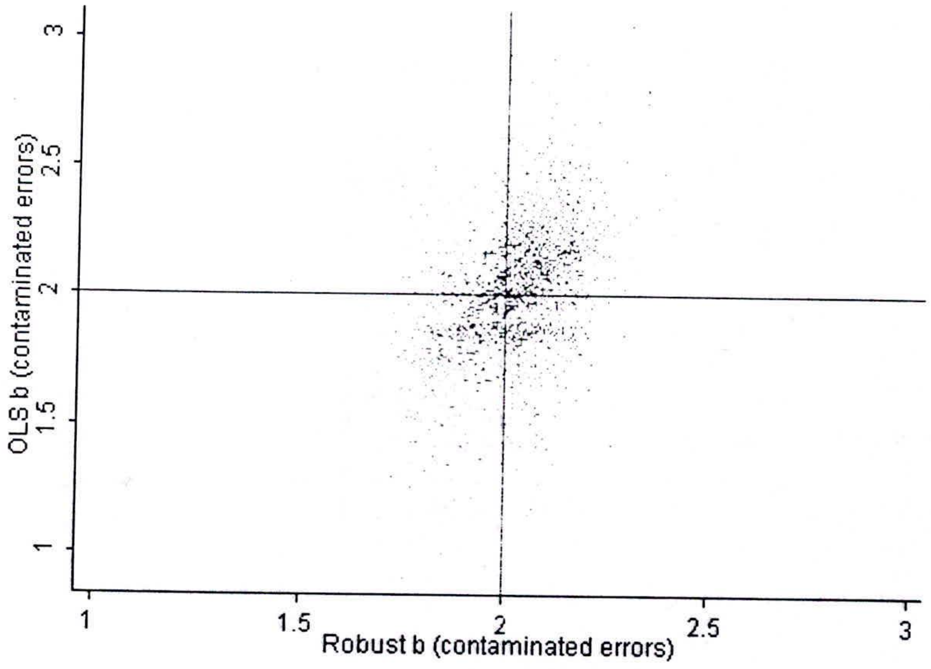
```
. quietly summarize b2

. global Varb2 = r(Var)

. quietly summarize b2r

. display 100*($Varb2/r(Var))
517.20057

. quietly summarize b2q

. display 100*($Varb2/r(Var))
328.3971
```

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 14.5. The variance of these OLS coefficients is more than five times greater than the variance of the corresponding robust coefficients, and more than three 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 14.6 illustrates the comparison between OLS and robust regression with a scatterplot showing 5,000 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(p) ylabel(1(.5)3, grid)
      yline(2) xlabel(1(.5)3, grid) xline(2)
```

Figure 14.6



The 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 small — less than 1%. For the theoretically-derived quantile standard errors the discrepancies appear a bit larger, between 3 and 4%. The least satisfactory estimates appear to be the bootstrapped quantile standard errors obtained by **bsqreg**. Means of the bootstrap standard errors exceed the observed standard deviation of *b1q* and *b2q* by 4 to 5%. Bootstrapping apparently over-estimated 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 how readily Stata supports Monte Carlo work.

References

Barron's Educational Series. 1992. *Barron's Çompact Guide to Colleges*, 8th ed. New York: Barron's Educational Series.

Beatty, J. Kelly, Brian O'Leary and Andrew 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, Lester R., William U. Chandler, Christopher Flavin, Cynthia Pollock, Sandra Postel, Linda Starke and Edward C. Wolf. 1986. *State of the World 1986*. New York: W. W. Norton.

Buch, E. 2000. *Oceanographic Investigations off West Greenland 1999*. Copenhagen: Danish Meteorological Institute.

CDC (Centers for Disease Control). 2003. Web site: http://www.cdc.gov

Chambers, John M., William S. Cleveland, Beat Kleiner and Paul A. Tukey (eds.). 1983. *Graphical Methods for Data Analysis*. Belmont, CA: Wadsworth.

Chatfield, C. 1996. *The Analysis of Time Series: An Introduction*, 5th edition. London: Chapman & Hall.

Chatterjee, S., A. S. Hadi and B. Price. 2000. *Regression Analysis by Example*, 3rd edition. New York: John Wiley & Sons.

Cleveland, William S. 1994. *The Elements of Graphing Data*. Monterey, CA: Wadsworth.

Cleves, Mario, William Gould and Roberto Gutierrez. 2004. *An Introduction to Survival Analysis Using Stata*, revised edition. College Station, TX: Stata Press.

Cook, R. Dennis and Sanford Weisberg. 1982. *Residuals and Influence in Regression*. New York: Chapman & Hall.

Cook, R. Dennis and Sanford Weisberg. 1994. *An Introduction to Regression Graphics*. New York: John Wiley & Sons.

Council on Environmental Quality. 1988. *Environmental Quality 1987–1988*. Washington, DC: Council on Environmental Quality.

Cox, C. Barry and Peter D. Moore. 1993. *Biogeography: An Ecological and Evolutionary Approach*. London: Blackwell Publishers.

Cryer, Jonathan B. and Robert B. Miller. 1994. *Statistics for Business: Data Analysis and Modeling*, 2nd edition. Belmont, CA: Duxbury Press.

Davis, Duane. 2000. *Business Research for Decision Making*, 5th edition. Belmont, CA: Duxbury Press.

DFO (Canadian Department of Fisheries and Oceans). 2003. Web site: http://www.meds-sdmm.dfo-mpo.gc.ca/alphapro/zmp/climate/IceCoverage_e.shtml

Diggle, P. J. 1990. *Time Series: A Biostatistical Introduction.* Oxford: Oxford University Press.

Efron, Bradley and R. Tibshirani. 1986. "Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy." *Statistical Science* 1(1):54–77.

Enders, W. 2003. *Applied Econometric Time Series*. 2nd edition. New York: John Wiley & Sons.

Everitt, Brian S., Savine Landau and Morven Leese. 2001. *Cluster Analysis*. 4th edition. London: Arnold.

Federal, Provincial, and Territorial Advisory Commission on Population Health. 1996. *Report on the Health of Canadians.* Ottowa: Health Canada Communications.

Fox, John. 1991. *Regression Diagnostics.* Newbury Park, CA: Sage Publications.

Fox, John and J. Scott Long. 1990. *Modern Methods of Data Analysis.* Beverly Hills: Sage Publications.

Frigge, Michael, David C. Hoaglin and Boris Iglewicz. 1989. "Some implementations of the boxplot." *The American Statistician* 43(1):50–54.

Gould, William, Jeffrey Pitblado and William Sribney. 2003. *Maximum Likelihood Estimation with Stata*, 2nd edition. College Station, TX: Stata Press.

Hamilton, Dave C. 2003. "The Effects of Alcohol on Perceived Attractiveness." Senior Thesis. Claremont, CA: Claremont McKenna College.

Hamilton, James D. 1994. *Time Series Analysis*. Princeton, NJ: Princeton University Press.

Hamilton, Lawrence C. 1985a. "Concern about toxic wastes: Three demographic predictors." *Sociological Perspectives* 28(4):463–486.

Hamilton, Lawrence C. 1985b. "Who cares about water pollution? Opinions in a small-town crisis." *Sociological Inquiry* 55(2):170–181.

Hamilton, Lawrence C. 1992a. *Regression with Graphics: A Second Course in Applied Statistics.* Pacific Grove, CA: Brooks/Cole.

Hamilton, Lawrence C. 1992b. "Quartiles, outliers and normality: Some Monte Carlo results." Pp. 92–95 in Joseph Hilbe (ed.) *Stata Technical Bulletin Reprints, Volume 1.* College Station, TX: Stata Press.

Hamilton, Lawrence C. 1996. *Data Analysis for Social Scientists.* Belmont, CA: Duxbury Press.

Hamilton, Lawrence C., Benjamin C. Brown and Rasmus Ole Rasmussen. 2003. "Local dimensions of climatic change: West Greenland's cod-to-shrimp transition." *Arctic* 56(3):271–282.

Hamilton, Lawrence C., Richard L. Haedrich and Cynthia M. Duncan. 2003. "Above and below the water: Social/ecological transformation in northwest Newfoundland." *Population and Environment* 25(2)101–121.

Hamilton, Lawrence C. and Carole L. Seyfrit. 1993. "Town-village contrasts in Alaskan youth aspirations." *Arctic* 46(3):255–263.

Hardin, James and Joseph Hilbe. 2001. *Generalized Linear Models and Extensions.* College Station, TX: Stata Press.

Hoaglin, David C., Frederick Mosteller and John W. Tukey (eds.). 1983. *Understanding Robust and Exploratory Data Analysis.* New York: John Wiley & Sons.

Hoaglin, David C., Frederick Mosteller and John W. Tukey (eds.). 1985. *Exploring Data Tables, Trends and Shape.* New York: John Wiley & Sons.

Hosmer, David W., Jr. and Stanley Lemeshow. 1999. *Applied Survival Analysis.* New York: John Wiley & Sons.

Hosmer, David W., Jr. and Stanley Lemeshow. 2000. *Applied Logistic Regression*, 2nd edition. New York: John Wiley & Sons.

Howell, David C. 1999. *Fundamental Statistics for the Behavioral Sciences*, 4th edition. Belmont, CA: Duxbury Press.

Howell, David C. 2002. *Statistical Methods for Psychology*, 5th edition. Belmont, CA: Duxbury Press.

Iman, Ronald L. 1994. *A Data-Based Approach to Statistics.* Belmont, CA: Duxbury Press.

Jentoft, Svein and Trond Kristoffersen. 1989. "Fishermen's co-management: The case of the Lofoten fishery." *Human Organization* 48(4):355–365.

Johnson, Anne M., Jane Wadsworth, Kaye Wellings, Sally Bradshaw and Julia Field. 1992. "Sexual lifestyles and HIV risk." *Nature* 360(3 December):410–412.

Johnston, Jack and John DiNardo. 1997. *Econometric Methods*, 4th edition. New York: McGraw-Hill.

Keller, Gerald, Brian Warrack and Henry Bartel. 2003. *Statistics for Management and Economics*, abbreviated 6th edition. Belmont, CA: Duxbury Press.

League of Conservation Voters. 1990. *The 1990 National Environmental Scorecard.* Washington, DC: League of Conservation Voters.

Lee, Elisa T. 1992. *Statistical Methods for Survival Data Analysis*, 2nd edition. New York: John Wiley & Sons.

Li, Guoying. 1985. "Robust regression." Pp. 281–343 in D. C. Hoaglin, F. Mosteller and J. W. Tukey (eds.) *Exploring Data Tables, Trends and Shape.* New York: John Wiley & Sons.

Long, J. Scott. 1997. *Regression Models for Categorical and Limited Dependent Variables.* Thousand Oaks, CA: Sage Publications.

Long, J. Scott and Jeremy Freese. 2003. *Regression Models for Categorical Outcomes Using Stata*, revised edition. College Station, TX: Stata Press.

MacKenzie, Donald. 1990. *Inventing Accuracy: A Historical Sociology of Nuclear Missile Guidance.* Cambridge, MA: MIT.

Mallows, C. L.. 1986. "Augmented partial residuals." *Technometrics* 28:313–319.

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, D. W. Jr. and J. A. Nelder. 1989. *Generalized Linear Models*, 2nd edition. London: Chapman & Hall.

Nash, James and Lawrence Schwartz. 1987. "Computers and the writing process." *Collegiate Microcomputer* 5(1):45–48.

National Center for Education Statistics. 1992. *Digest of Education Statistics 1992*. Washington, DC: U.S. Government Printing Office.

National Center for Education Statistics. 1993. *Digest of Education Statistics 1993*. Washington, DC: U.S. Government Printing Office.

Newton, H. Joseph and Jane L. Harvill. 1997. *StatConcepts: A Visual Tour of Statistical Ideas*. Pacific Grove, CA: Duxbury Press.

Pagano, Marcello and Kim Gauvreau. 2000. *Principles of Biostatistics*, 2nd edition. Belmont, CA: Duxbury Press.

Rabe–Hesketh, Sophia and Brian Everitt. 2000. *A Handbook of Statistical Analysis Using Stata*, 2nd edition. Boca Raton, FL: Chapman & Hall.

Report of the Presidential Commission on the Space Shuttle Challenger Accident. 1986. Washington, DC.

Rosner, Bernard. 1995. *Fundamentals of Biostatistics*, 4th edition. Belmont, CA: Duxbury Press.

Selvin, Steve. 1995. *Practical Biostatistical Methods*. Belmont, CA: Duxbury Press.

Selvin, Steve. 1996. *Statistical Analysis of Epidemiologic Data*, 2nd edition. New York: Oxford University.

Seyfrit, Carole L.. 1993. *Hibernia's Generation: Social Impacts of Oil Development on Adolescents in Newfoundland*. St. John's: Institute of Social and Economic Research, Memorial University of Newfoundland.

Shumway, R. H. 1988. *Applied Statistical Time Series Analysis*. Upper Saddle River, NJ: Prentice–Hall.

Stata Corporation. 2005. *Getting Started with Stata for Macintosh*. College Station, TX: Stata Press.

Stata Corporation. 2005. *Getting Started with Stata for Unix*. College Station, TX: Stata Press.

Stata Corporation. 2005. *Getting Started with Stata for Windows*. College Station, TX: Stata Press.

Stata Corporation. 2005. *Mata Reference Manual*. College Station, TX: Stata Press.

Stata Corporation. 2005. *Stata Base Reference Manual* (3 volumes). College Station, TX: Stata Press.

Stata Corporation. 2005. *Stata Data Management Reference Manual.* College Station, TX: Stata Press.

Stata Corporation. 2005. *Stata Graphics Reference Manual.* College Station, TX: Stata Press.

Stata Corporation. 2005. *Stata Programming Reference Manual.* College Station, TX: Stata Press.

Stata Corporation. 2005. *Stata Longitudinal/Panel Data Reference Manual.* College Station, TX: Stata Press.

Stata Corporation. 2005. *Stata Multivariate Statistics Reference Manual.* College Station, TX: Stata Press.

Stata Corporation. 2005. *Stata Quick Reference and Index.* College Station, TX: Stata Press.

Stata Corporation. 2005. *Stata Survey Data Reference Manual.* College Station, TX: Stata Press.

Stata Corporation. 2005. *Stata Survival Analysis and Epidemiological Tables Reference Manual.* College Station, TX: Stata Press.

Stata Corporation. 2005. *Stata Time-Series Reference Manual.* College Station, TX: Stata Press.

Stata Corporation. 2005. *Stata User's Guide.* College Station, TX: Stata Press.

Stine, Robert and John Fox (eds.). 1997. *Statistical Computing Environments for Social Research.* Thousand Oaks, CA: Sage Publications.

Topliss, Brenda J. 2001. "Climate variability I: A conceptual approach to ocean–atmosphere feedback." In Abstracts for AGU Chapman Conference, The North Atlantic Oscillation, Nov. 28 – Dec. 1, 2000, Ourense, Spain.

Tufte, Edward R. 1997. *Visual Explanations: Images and Quantities, Evidence and Narratives.* Cheshire, CT: Graphics Press.

Tukey, John W. 1977. *Exploratory Data Analysis.* Reading, MA: Addison–Wesley.

Velleman, Paul F. 1982. "Applied Nonlinear Smoothing," pp.141–177 in Samuel Leinhardt (ed.) *Sociological Methodology 1982.* San Francisco: Jossey-Bass.

Velleman, Paul F. and David C. Hoaglin. 1981. *Applications, Basics and Computing of Exploratory Data Analysis.* Boston: Wadsworth.

Ward, Sally and Susan Ault. 1990. "AIDS knowledge, fear, and safe sex practices on campus." *Sociology and Social Research* 74(3):158–161.

Werner, Al. 1990. "Lichen growth rates for the northwest coast of Spitsbergen, Svalbard." *Arctic and Alpine Research* 22(2):129–140.

World Bank. 1987. *World Development Report 1987.* New York: Oxford University.

World Resources Institute. 1993. *The 1993 Information Please Environmental Almanac.* Boston: Houghton Mifflin.

Index

A

ac (autocorrelations), 339, 351–352
acprplot (augmented component-plus-residual plot), 197, 202–203
added-variable plot, 198, 201–202
ado-file (automatic do), 233–235, 362, 373–375
alpha (Cronbach's alpha reliability), 318–319
analysis of covariance (ANCOVA), 141–142, 153–154
analysis of variance (ANOVA)
  factorial, 142, 152–153, 156
  interaction effects, 142, 152–154, 156–157
  median, 253–255
  N-way, 152–153
  one-way, 142, 155
  predicted values, 155–158, 167
  regression model, 153–154, 249–256
  repeated-measures, 142
  robust, 249–256
  standard errors, 155–157, 167
  three-way, 142
  two-way, 142, 152–153, 156–157
anova, 142, 152–158, 167, 239
append, 13, 42–44
ARCH model (autoregressive conditional heteroskedasticity), 339
area plot, 86–87
args (arguments in program), 366–368
areg (absorb variables in regression), 179–180
ARIMA model (autoregressive integrated moving average), 339, 354–360
arithmetic operator, 26
artificial data, 14, 57–61, 241, 387–394
ASCII (text) file
  read data, 13–14, 39–42
  write data, 42
  write results (log file), 2–3, 6–7
autocode (create ordinal variables), 31, 37–38
autocorrelation, 339, 350–352, 357–358, 369–373
aweight (analytical weights), 54
axis label in graph, 66
  angle, 81–82
  format, 13, 24–25, 76, 305–306
  grid, 113–115
  suppress, 118, 129, 173
axis scale in graph, 66, 112–118

B

_b coefficients (regression), 230, 269, 273–274, 285, 356
band regression, 217–219
bar chart, 94–99, 147, 150–151
Bartlett's test for equal variances, 149–150
batch-mode program, 61
bcskew0 (transform to reduce skew), 129
beta weight (standardized regression coefficient), 160, 164–165
Bonferroni multiple-comparison test
  correlation matrix, 172–173
  one-way ANOVA, 150–151
bootstrap, 246, 315–316, 382–387, 389–394
box plot, 66, 90–91, 118–119, 147, 150–151, 389, 392

Box–Cox
  regression, 215, 226–227
  transformation, 129
Box–Pierce *Q* test (white noise), 341, 351, 354, 357–358
browse (Data Browser), 13
bs (bootstrap), 382–387
bsqreg (quantile regression with bootstrap), 240, 246, 389–394
by prefix, 121, 133–134

C

c chart (quality control), 105
caption in graph, 109–110
case identification number, 38–39
categorical variable, 35–39, 183–185
censored-normal regression, 264
centering to reduce multicollinearity, 212–214
chi-squared
  deviance (logistic regression), 271, 275–278
  equal variances in ANOVA, 149–150
  independence in cross-tabulation, 55, 130–133, 281
  likelihood-ratio in cross-tabulation, 130–131, 281
  likelihood-ratio in logistic regression, 267–268, 270, 272–273, 281
  probability plot, 105
  quantile plot, 105
ci (confidence interval), 124, 255
cii (immediate confidence interval), 124
classification table (logistic regression), 264, 270–272
clear (remove data from memory), 14–15, 23, 362
cluster analysis, 318–320, 329–338
coefficient of variation, 123–124
collapse, 52–53
color
  bar chart, 95–96
  pie chart, 92
  scatterplot symbols, 74
  shaded regions, 86
combine data files, 14, 42–47
combine graphs. *See* **graph combine**
comments in programs, 364, 369–370, 373–374
communality (factor analysis), 326
component-plus-residual plot, 197–198, 202–203
compress, 13, 40, 60–61
conditional effect plot, 230–232, 273–274, 284–287
confidence interval
  binomial, 124
  bootstrap, 383–384, 386
  mean, 124
  regression coefficients, 163
  regression line, 66, 85, 110–112, 160
  robust mean, 255
  Poisson, 124
constraint (linear constraints), 262
Cook and Weisberg heteroskedasticity test, 197
Cook's *D*, 158, 167, 197, 206–210
copy results, 4
correlation
  hypothesis test, 160, 172–173
  Kendall's tau, 131, 174–175
  matrix, 18, 59, 160, 171–174
  Pearson product-moment, 1, 18, 160, 171–173
  regression coefficient estimates, 214
  Spearman, 174
corrgram (autocorrelation), 339, 351, 357–358, 373
count-time data, 293–295
covariance
  regression coefficient estimates, 167, 173, 197, 214
  variables, 160, 173
COVRATIO, 167, 197, 206
Cox proportional hazard model, 290, 299–305
Cramer's *V*, 131
Cronbach's alpha, 318–319
cross-correlation, 353–354
cross-tabulation, 121, 130–136
ctset (define count-time data), 289, 293–294

cttost (convert count-time to survival-time data), 289, 294–295
cubic spline curve. *See* **graph twoway mspline**
cv (coefficient of variation), 123–124

D

Data Browser, 13
data dictionary, 41
Data Editor, 13, 15–16
data management, 12–63
database file, 41–42
date, 30, 266, 340–342
decode (numeric to string), 33–34
#delimit (end-of-line delimiter), 61, 116, 362
dendrogram, 319, 329, 331–337
describe (describe data), 3, 18
destring (string to numeric), 35
DFBETA, 158, 167, 197, 205–206, 208–210
DFITS, 167, 197, 206, 208–210
diagnostic statistics
  ANOVA, 158, 167
  logistic regression, 271, 274–278
  regression, 167, 196–214
Dickey–Fuller test, 340, 355–356
difference (time series), 349–350
display (show value onscreen), 31–32, 39, 211, 269
display format, 13, 24–25, 359
do-file, 60–61, 115–116, 361–362, 367–373
Do-File Editor, 60, 361
dot plot, 67, 95, 99–100, 150–151
drawnorm (normal variable), 13, 59
drop
  variable in memory, 22
  data in memory, 14–15, 23, 40, 56
  program in memory, 363, 373–375
dummy variable, 35–36, 176–185, 267
Durbin–Watson test, 158, 197, 350
dwstat (Durbin–Watson test), 197, 350

E

e-class, 381, 386
edit (Data Editor), 13, 15–16
effect coding for ANOVA, 250–251
efficiency of estimator, 393
egen, 33, 331, 340, 343
eigenvalue, 318–319, 321, 326
empirical orthogonal function (EOF), 325
Encapsulated Postscript (.eps) graph, 6, 116
encode (string to numeric), 13, 33–34
epidemiological tables, 288
error-bar plot, 143, 155–157
estimates store (hypothesis testing), 272–273, 278–279, 282–283
event-count model, 288, 290, 310–313
Exploratory Data Analysis (EDA), 124–126
exponential filter (time series), 343
exponential growth model, 216, 232–235
exponential regression (survival analysis), 305–307

F

factor analysis, 318–328
factor rotation, 318–319, 322–325
factor score, 318–319, 323–325
factorial ANOVA, 142, 152–153, 156
FAQs (frequently asked questions), 8
filter, 343
Fisher's exact test in cross-tabulation, 131
fixed and random effects, 162
foreach, 365
format
  axis label in graph, 76, 305–306
  input data, 40–41
  numerical display, 13, 24–25, 359
forvalues, 365
frequency table, 130–133, 138–139
frequency weights, 54–55, 66, 73–74, 120, 123, 138–140
function
  date, 30
  mathematical, 27–28
  probability, 28–30
  special, 31
  string, 31
fweight (frequency weights), 54–55, 73–74, 138–140

G

generalized linear modeling (GLM), 264, 291, 313–317
generate, 13, 23–26, 37, 39
gladder, 128
Gompertz growth model, 234–238
Goodman and Kruskal's gamma, 131
graph bar, 66–67, 94–99, 147
graph box, 66, 90–91, 118–119, 147, 389, 392
graph combine, 117–119, 147, 150–151, 222, 231–232
graph dot, 67, 95, 99–100, 150–151
graph export, 116
graph hbar, 97–98
graph hbox, 91, 150–151
graph matrix, 66, 77, 173–174
graph pie, 66, 92–94
graph twoway
  all types, 84–85
  overlays, 66, 85, 110–115, 344–345, 347–348
graph twoway area, 84, 85–86
graph twoway bar, 84
graph twoway connected, 5–6, 50–51, 66, 79–80, 83–84, 114–115, 157, 192–193
graph twoway dot, 85
graph twoway lfit, 66, 74, 85, 110, 168, 181
graph twoway lfitci, 85, 110–112, 170–171
graph twoway line, 66, 77–82, 112–115, 117, 221–222, 242, 244, 247, 344–345, 371
graph twoway lowess, 85, 88–89, 216, 219–221
graph twoway mband, 85, 216, 217–219
graph twoway mspline, 85, 182, 190, 218–219, 226, 287–288
graph twoway qfit, 85, 110, 190
graph twoway rarea, 84, 170
graph twoway rbar, 85
graph twoway rcap, 85, 89, 157
graph twoway scatter, 65–66, 72–77, 181–182, 277, 394
graph twoway spike, 84, 87–88, 347
graph use, 116
graph7, 65
gray scale, 86
greigen (graph eigenvalues), 318–319, 321–322
gsort (general sorting), 14

H

hat matrix, 167, 205–206, 210
hazard function, 290, 302, 307, 309
help, 7
help file, 7, 375–377
heteroskedasticity, 161, 197, 199–200, 223–224, 239, 256–258, 290, 315, 339
hettest (heteroskedasticity test), 197, 199–200
hierarchical linear models, 162
histogram, 65, 67–71, 385
Holt–Winters smoothing, 343
Huber/White robust standard errors, 160, 256–261

I

if qualifier, 13, 14, 19–23, 204–205, 209
if...else, 366
import data, 39–42
in qualifier, 14, 19–23, 166
incidence rate, 289–290, 293, 297, 309–310, 312
inequality, 21
infile (read ASCII data), 13–14, 40–42
infix (read fixed-format data), 41–42
influence
  logistic regression, 271, 274–278
  regression (OLS), 167, 196–198, 201, 204–208
  robust regression, 248
insert
  graph into document, 6
  table into document, 4
insheet (read spreadsheet data), 41–42
instrumental variables (2SLS), 161

interaction effect
  ANOVA, 142, 152–157, 250–253
  regression, 160. 180–185, 211–212, 259–261
interquartile range (IQR). 53, 91, 95, 103, 123–124, 126, 135
iteratively reweighted least squares (IRLS), 242
iweight (importance weights). 54

J
jackknife
  residuals, 167
  standard errors, 314–317

K
Kaplan–Meier survivor function, 289–290, 295–298
keep (keep variable or observation), 23, 173
Kendall's tau, 131, 174–175
kernel density, 65, 70, 85
Kruskal–Wallis test, 142, 151–152
kurtosis, 122–124, 126–127

L
L-estimator, 243
label data, 18
label define, 26
label values, 25–26
label variable, 16, 18
ladder of powers, 127–129
lag (time series), 349–350
lead (time series), 349–350
legend in graph, 78, 81, 112, 114–115, 157, 221, 344
letter-value display, 125–126
leverage, 158, 159, 167, 196, 198, 201–206, 210, 229, 246–248
leverage-vs.-squared-residuals plot, 198, 203–204
lfit (fit of logistic model), 264
likelihood-ratio chi-squared. *See* chi-squared
line in graph
  pattern, 81–82, 84, 115
  width, 221, 344, 371
line plot, 77–84
link function (GLM), 291, 313–317
list, 3–4, 14, 17, 19, 49, 54, 265
log, 2–3
log file, 2–3, 6–7
logarithm, 27, 127–129, 223–229
logical operator, 20
logistic growth model, 216, 233–234
logistic regression, 262–287
logistic (logistic regression), 185, 262–264, 269–278
logit (logistic regression), 267–269
looping, 365–366
lowess smoothing, 88–89, 216, 219–222
lroc (logistic ROC), 264
lrtest (likelihood-ratio test), 272–273, 278–279, 282–283
lsens (logistic sensitivity graph), 264
lstat (logistic classification table), 264, 270–272
lvr2plot (leverage-vs.-squared-residuals plot), 198, 203–204

M
M-estimator, 243
macro, 235, 334, 363, 365, 367, 370, 387
Mann–Whitney *U* test, 142, 148–149, 152
margin in graph, 110, 113, 117–118, 192–193
marker label in graph, 66, 75–76, 202, 204
marker symbol in graph, 66, 73–75, 84, 100, 183, 277
marksample, 368–369
matched-pairs test, 143, 145–146
matrix algebra, 378–382
mean, 122–124, 126,135–137, 139–140, 143–158, 387–389
median, 90–91, 122–124, 126, 135–137, 387–389
median regression. *See* quantile regression
memory, 14, 61–63
merge, 14, 44–50
missing value, 13–16, 21, 37–38

Monte Carlo, 126, 246, 387–394
moving average
  filter, 340, 343–344
  time series model, 354–360
multicollinearity, 210–214
multinomial logistic regression, 264, 278, 280–287
multiple-comparison test
  correlation matrix, 172–173
  one-way ANOVA, 150–151

N

negative exponential growth model, 233
nolabel option, 32–34
nonlinear regression, 216, 232–238
nonlinear smoothing, 340–341, 343–346
normal distribution
  artificial data, 13, 59, 241
  curve, 65
  test for, 126–129
normal probability plot. *See* quantile–normal plot
numerical variables, 16, 20, 122

O

OBDC (Open Database Connectivity), 42
odds ratio. *See* logistic regression
observation number, 38–39
omitted-variables test, 197, 199
one-sample *t* test, 143–146
one-way ANOVA, 149–152
open file, 2
order (order variables in data), 19
ordered logistic regression, 278–280
ordinal variable, 35–36
outfile (write ASCII data), 42
outlier, 126, 239–248, 344, 388–394
overlay twoway graphs, 110–115

P

p chart (quality control), 105–107
paired-difference test, 143, 145–146
panel data, 161, 191–195
partial autocorrelation, 339–340, 352
partial regression plot. *See* added-variable plot
Pearson correlation, 5, 19, 160, 171–173
percentiles, 122–124, 136
periodogram, 340
Phillips–Perron test, 355
pie chart, 66, 92–94
placement (legend in graph), 114–115
poisgof (Poisson goodness of fit test), 310–311
Poisson regression, 290–291, 309–313, 317
polynomial regression, 188–191
Portable Network Graphics (.png) graph, 6, 116
Postscript (.ps or .eps) graph, 6, 116
Prais–Winsten regression, 340, 359–360
predict (predicted values, residuals, diagnostics)
  anova, 155–158, 167
  arima, 357
  logistic, 264, 268–271, 284
  regress, 159, 165–167, 190, 196–197, 205–210, 216, 233
principal components, 318–325
print graph, 6
print results, 4
probit regression, 262–263, 314
program, 362–363
promax rotation, 319, 322–325
pweight (probability or sampling weights), 54–56
pwcorr (pairwise Pearson correlation), 160, 172–173, 174–175

Q

qladder, 128–129
quality-control graphs, 67, 105–108
quantile
  defined, 102
  quantile plot, 102–103
  quantile–normal plot, 67, 104
  quantile–quantile plot, 104–105
  regression, 239–256, 389–394
quartile, 91, 125–126
quietly, 175, 182, 188

R

r chart (quality control), 67, 106, 108

r-class, 381, 387, 390
Ramsey specification error test (RESET), 197
random data, 56–60, 241, 387–394
random number, 30, 56–59, 241
random sample, 14, 60
range (create data over range), 236
range plot, 89
range standardization, 334–335
rank, 32
rank-sum test, 142, 148–149, 152
real function, 35–36
regress (linear regression), 159–165, 239, 386, 389–394
regression
  absorb categorical variable, 179–180
  beta weight (standardized regression coefficient), 160, 164–165
  censored-normal, 264
  confidence interval, 110–112, 163, 169–171
  constant, 163
  curvilinear, 189–191, 216, 223–232
  diagnostics, 167, 196–214
  dummy variable, 176–185
  hypothesis test, 160, 175–176
  instrumental variable, 161
  line, 67, 110–112, 159–160, 168–171, 190, 242, 244, 247
  logistic, 262–287
  multinomial logistic, 264, 278, 280–287
  multiple, 164–165
  no constant, 163
  nonlinear, 232–238
  ordered logistic, 278–280
  ordinary least squares (OLS), 159–165
  Poisson, 290–291, 309–313, 317
  polynomial, 188–191
  predicted value, 165–167, 169
  probit, 262–263, 314
  residual, 165–167, 169, 205–207
  robust, 239–256, 389–394
  robust standard errors, 256–261
  stepwise, 161, 186–188
  tobit, 188, 263
  transformed variables, 189–191, 216, 223–232
  two-stage least squares (2SLS), 161
  weighted least squares (WLS), 161, 245
relational operator, 20
relative risk ratio, 264, 281–284
rename, 16, 17
replace, 16, 25–26, 33
RESET (Ramsey test), 197
reshape, 49–52
residual, 159–160, 167, 200–208
residual-vs.-fitted (predicted values) plot, 160, 169, 188–191, 198, 200
retrieve graph, 116
robust
  anova, 249–255
  mean, 255
  regression, 239–256
  standard errors and variance, 256–261
ROC curve (receiver operating characteristic), 264
rotation (factor analysis), 318–319, 322–325
rough, 345
rreg (robust regression), 239–256, 389–394
rvfplot (residual-vs.-fitted plot), 160, 188–191, 198, 200
rvpplot (residual-vs.-predictor plot

S

sample (draw random sample), 14, 60
sampling weights, 55–56
sandwich estimator of variance, 160, 256–261
SAS data files, 42
save (save dataset), 14, 16, 23
save graph, 6
saveold (save dataset in previous Stata format), 14
scatterplot. *Also see* **graph twoway scatter**
  axis labels, 66, 72
  basic, 66–67
  marker labels, 67, 74–75, 202–204

marker symbols, 72–73, 119, 182–183
matrix, 66, 77, 173–174
weighting, 66, 74–75, 207–208
with regression line, 66, 110–112, 159–160, 181–182
Scheffé multiple-comparison test, 150–151
score (factor scores), 318–319, 323–325
scree graph (eigenvalues), 318–319, 321–322
search, 8–9
seasonal difference (time series), 349–350
serrbar (standard-error bar plot), 143, 155–157
set memory, 14, 62–63
shading
color, 86
intensity, 91
Shapiro–Francia test, 127
Shapiro–Wilk test, 127
shewart, 106
Šidák multiple-comparison test, 150, 172–173
sign test, 144–145
signed-rank test, 143 146
skewness, 122–124, 126–127
sktest (skewness–kurtosis test), 126–127, 383
slope dummy variable, 180
SMCL (Stata Markup and Control Language), 376–377
smoothing, 340–341, 343–346
sort, 14, 19, 21–22, 166
Spearman correlation, 174–175
spectral density, 340
spike plot, 84, 87–88, 347
spreadsheet data, 41–42
SPSS data files, 42
standard deviation, 122–124, 126, 135
standard error
ANOVA, 155–157
bootstrap. *See* bootstrap
mean, 124
regression prediction, 167, 169–171
robust (Huber/White), 160, 256–261
standardized regression coefficient, 160, 164–165
standardized variable, 32, 331
Stat/Transfer, 42
Stata Journal, 10–11
Statalist online forum, 10
stationary time series, 340, 355–356
stcox (Cox hazard model), 290, 299–303
stcurve (survival analysis graphs), 290, 307
stdes (describe survival-time data), 289, 292–293
stem-and-leaf display, 124–125
stepwise regression, 161, 186–188
stphplot, 290
streg (survival-analysis regression), 290, 305–309
string to numeric, 32–35
string variable, 17, 40–41
sts generate (generate survivor function), 290
sts graph (graph survivor function), 289, 296, 298
sts list (list survivor function), 290
sts test (test survivor function), 290, 298
stset (define survival-time data), 289, 291–292, 297
stsum (summarize survival-time data), 289, 293, 297
studentized residual, 167, 205, 207
subscript, 39–40, 343
summarize (summary statistics), 2, 17, 20, 31–32, 90–91, 120–124, 383
sunflower plot, 74–75
survey sampling weights, 55–56, 161, 263
survival analysis, 288–309
svy: regress (survey data regression), 161
svyset (survey data definition), 56
sw (stepwise model fitting), 186–188
symmetry plot, 100, 102

syntax (programming), 368–369

T

t test
  correlation coefficient, 160, 172–173
  means, 143–149
  robust means, 255
  unequal variance, 148
table, 121, 134–136, 152
tabstat, 120, 123–124
tabulate, 4, 15, 36–37, 56, 121, 130–133, 136
technical support, 9
test (hypothesis test for model), 160, 175–176, 312
text in graph, 109–110, 113, 222
time plot, 77–84, 343–348
time series, 339–360
tin (times in), 346–347, 350, 359
title in graph, 109–110, 112–113
tobit regression, 188, 263
transfer data, 42
transform variable, 126–129, 189–190, 216
transpose data, 47–49
tree diagram, 319, 329, 331–337
tsset (define time series data), 340, 342, 346
tssmooth (time series smoothing), 340–341, 343–346
ttest, 143–149, 392
Tukey, John, 124
twithin (times within), 346–347
two-sample test, 146–149
two-stage least squares (2SLS), 161

U

unequal variance in *t* test, 143, 148–149
uniform (random number generator), 30, 56–58, 241
unit root, 355–356
use, 2–3, 15

V

variance, 122–124, 135, 214
variance inflation factor, 197, 211–212
varimax rotation, 319, 322–325
version, 364

W

web site, 9
Weibull regression (survival analysis), 305, 307–399
weighted least squares (WLS), 161, 245
weights, 55–57, 74–75, 122–124, 138–140, 161
Welsch's distance, 167, 206–210
which, 374
while, 365–366
white noise, 341, 351, 354, 357–358
Wilcoxon rank-sum test, 142, 148–149, 152
Wilcoxon signed-rank test, 143, 146
Windows metafile (.wmf or .emf) graph, 6, 116
wntest (Box–Pierce white noise *Q* test), 341
word processor
  insert Stata graph into, 6
  insert Stata table into, 4

X

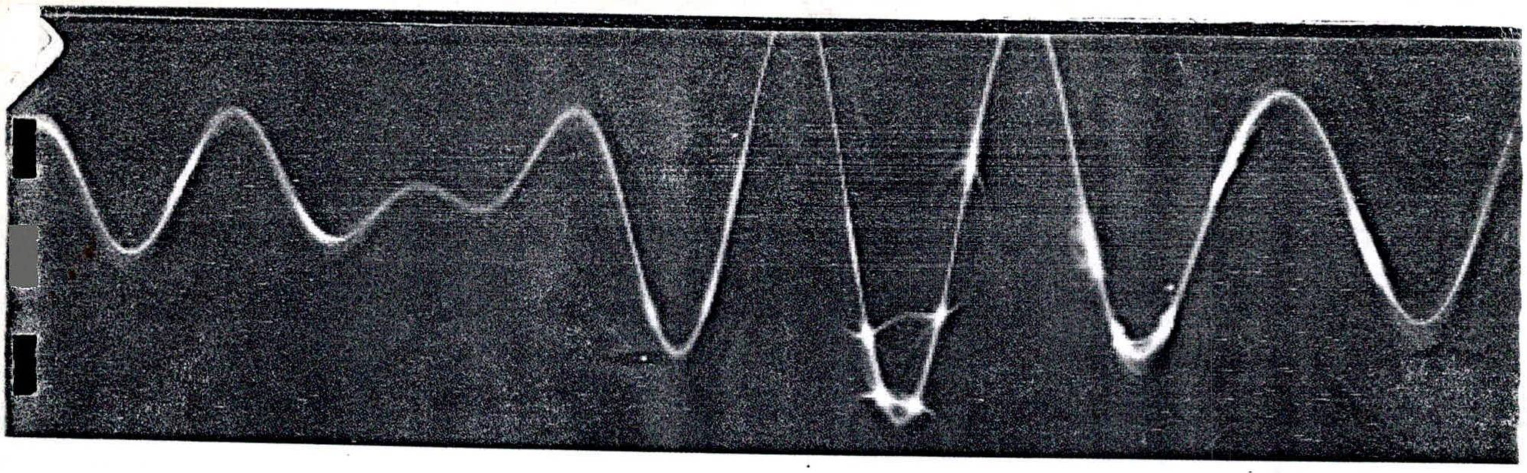
x axis in graph. *See* axis label in graph, axis scale in graph
x-bar chart (quality control), 106–108
xcorr (cross-correlation), 353–354
xi (expanded interaction terms), 160, 183–185
xpose (transpose data), 48–49
xtmixed (multilevel mixed-effect models), 162
xtreg (panel data regression), 161, 191–195

Y

y axis in graph. *See* axis label in graph, axis scale in graph

Z

z score (standardized variable), 32, 331



Your Guide to a Powerful, State-of-the-Art Statistical Program— Now updated for use with Version 9!

For students and practicing researchers alike, *Statistics with Stata* opens the door to full use of the popular *Stata* program—a fast, flexible, and easy-to-use environment for data management and statistical analysis. Now integrating *Stata's* impressive new graphics, this comprehensive book presents hundreds of examples showing how you can apply *Stata* to accomplish a wide variety of tasks. Like *Stata* itself, *Statistics with Stata* will make it easier for you to move fluidly through the world of modern data analysis. Its contents include:

- ▲ A complete chapter on database management, including sections on how to create, translate, update, or restructure datasets.
- ▲ A detailed, example-based introduction to the new graphical capabilities of *Stata*. Topics range from simple histograms and time plots to regression diagnostics and quality control charts. New sections describe methods to combine or enhance graphs for publication.
- ▲ Basic statistical tools, including tables, parametric tests, chi-square and other nonparametric tests, *t* tests, ANOVA/ANCOVA, correlation, linear regression, and multiple regression.
- ▲ Advanced methods, including nonlinear, robust, and quantile regression; logit, multinomial logit, and other models for categorical dependent variables; survival and event-count analysis; generalized linear modeling (GLM), factor analysis, and cluster analysis—all demonstrated through practical, easy-to-follow examples with an emphasis on interpretation.
- ▲ Guidelines for writing your own programs in *Stata*—user-written programs allow creation of powerful new tools for database management and statistical analysis and support computation-intensive methods, such as bootstrapping and Monte Carlo simulation.

Data files are available at **http://www.duxbury.com**, the Duxbury Web site.

THOMSON
BROOKS/COLE

Visit Brooks/Cole online at **www.brookscole.com**

For your learning solutions: **www.thomson.com/learning**

ISBN 049510972X

9 780495 109723

STATISTICS WITH STATA VERS 9 5TH 06 ITP PB HAMILTON

9 780495 109723