STATA TECHNICAL BULLETIN

September 1993

STB-15

A publication to promote communication among Stata users

Contents of this issue | page

		page
an1.1.	STB categories and insert codes (Reprint)	2
an36.	Stata 3.0 users beware!	2
crc32.	Correction to 3.1 manual description of `S_MACH`	2
crc33.	Linear spline construction	2
dt1.	Five data sets for teaching	4
gr13.	Incorporating Stata-created PostScript files into TEX/LATEX documents	7
gr13.1.	`\special{}` effects with Stata graphs in TEX documents	12
sg19.	Linear splines and piecewise linear functions	13
sss1.	Calculating U.S. marginal income tax rates	17
sts4.	A suite of programs for time series regression	20

an1.1 STB categories and insert codes

Inserts in the STB are presently categorized as follows:

General Categories:			
an	announcements	*ip*	instruction on programming
cc	communications & letters	*os*	operating system, hardware, & interprogram communication
dm	data management		
dt	data sets	*qs*	questions and suggestions
gr	graphics	*tt*	teaching
in	instruction	*zz*	not elsewhere classified
Statistical Categories:			
sbe	biostatistics & epidemiology	*srd*	robust methods & statistical diagnostics
sed	exploratory data analysis	*ssa*	survival analysis
sg	general statistics	*ssi*	simulation & random numbers
smv	multivariate analysis	*sss*	social science & psychometrics
snp	nonparametric methods	*sts*	time-series, econometrics
sqc	quality control	*sxd*	experimental design
sqv	analysis of qualitative variables	*szz*	not elsewhere classified

In addition, we have granted one other prefix, *crc*, to the manufacturers of Stata for their exclusive use.

an36 Stata 3.0 users beware!

Sean Becketti, Editor, STB, FAX 913-888-6708

Many of the programs published in this issue of the STB are written for Stata 3.1, not 3.0. In particular, Stata 3.0 users should *not* install the *crc* updates published in this issue. For programs in other sections, I indicate at the top whether they will work with Stata 3.0. Any program that will work with Stata 3.0 will, of course, work with Stata 3.1.

crc32 Correction to 3.1 manual description of S_MACH

The Stata 3.1 manual ([2] macro) documents a new system macro, `S_MACH`, that describes the type of computer currently running Stata. The manual incorrectly calls this macro `S_MACHID`. The correct name of this macro is `S_MACH`. The rest of the description in the manual is correct.

crc33 Linear spline construction

[`mkspline` is for use with Stata 3.1; see an36. Also see sg19 in this issue for more discussion on linear splines and piecewise linear functions.—Ed.]

The syntax of `mkspline` is

`mkspline` *newvar*$_1$ #$_1$ [*newvar*$_2$ #$_2$ [...]] *newvar*$_k$ = *oldvar* [**`if`** *exp*] [**`in`** *range*] [, **`marginal`**]

or

`mkspline` *stubname* # = *oldvar* [**`if`** *exp*] [**`in`** *range*] [, **`marginal`** **`pctile`**]

In the first syntax, `mkspline` creates *newvar*$_1$, ..., *newvar*$_k$ containing a linear spline of *oldvar* with knots at the specified #$_1$, ..., #$_{k-1}$.

In the second syntax, `mkspline` creates # variables named *stubname*`1`, *stubname*`2`, ..., containing a linear spline of *oldvar*. The knots are equally spaced over the range of *oldvar* or are placed at the percentiles of *oldvar*.

Options

`marginal`, which may be used with either syntax, specifies that the new variables are to be constructed so that, when used in estimation, the coefficients represent the change in the slope from the preceding interval. The default is to construct the variables so that, when used in estimation, the coefficients will measure the slopes for the interval. For example, consider estimating the regression

$$y = a_1\,\texttt{age1} + a_2\,\texttt{age2} + a_3\,\texttt{age3} + \mathbf{X}\mathcal{B} + u$$

after ‘`mkspline age1 20 age2 30 age3 30 = age`’. The interpretation of the coefficients is

$$\frac{dy}{d\texttt{age}} = \begin{cases} a_1 & \text{if } \texttt{age} < 20 \\ a_2 & \text{if } 20 \le \texttt{age} < 30 \\ a_3 & \text{otherwise.} \end{cases}$$

After ‘`mkspline age1 20 age2 30 age3 30 = age, marginal`’, however, the interpretation would be

$$\frac{dy}{d\texttt{age}} = \begin{cases} a_1 & \text{if } \texttt{age} < 20 \\ a_1 + a_2 & \text{if } 20 \le \texttt{age} < 30 \\ a_1 + a_2 + a_3 & \text{otherwise.} \end{cases}$$

`pctile` is allowed only with the second syntax. It specifies that the knots are to be placed at percentiles of the data rather than equally spaced based on the range. For instance,

`mkspline m 2 = mpg`

creates new variables `m1` and `m2`. If `mpg` ranged from 10 to 40, the knot would be placed at the midpoint of the range, 25.

`mkspline m 2 = mpg, pctile`

would also create `m1` and `m2`, but the knot would be placed at the median value of `mpg`. Changing the `2` to a `3` in each of the examples above, three variables would be created: `m1`, `m2`, and `m3`. Without the `pctile` option, the range would be split into thirds: 10–20, 20–30, and 30–40 (the knots would be at 20 and 30). With the `pctile` option, the knots would be placed at the 33rd and 66th percentiles of `mpg`, thus splitting the data itself into thirds.

Remarks

You wish to estimate the model

$$\texttt{lninc} = b_0 + b_1\,\texttt{educ} + f(\texttt{age}) + u$$

using a piecewise linear function for age. The knots are to be at ten-year intervals: 20, 30, 40, 50, and 60.

```
. mkspline age1 20 age2 30 age3 40 age4 50 age5 60 age6 = age
. regress lninc educ age1-age6
  (output omitted)
```

To test that the age effect is the same in the 30–40 and 40–50 intervals, type

```
. test age3 = age4
oom
```

Alternatively, had you specified the `marginal` option on the `mkspline` command,

```
. mkspline age1 20 age2 30 age3 40 age4 50 age5 60 age6 = age, marginal
. regress lninc educ age1-age6
oom
```

you would test whether the age effect is the same in the 30–40 and 40–50 intervals by asking whether the `age4` coefficient were zero. With the `marginal` option, coefficients measure the change in slope from the preceding group. Specifying `marginal` changes only the interpretation of the coefficients; the same model is estimated in either case.

As a second example, pretend you have a binary outcome variable called `outcome`. You are beginning an analysis and wish to parameterize the effect of dosage on outcome. You wish to divide the data into 5-equal width groups of dosage for the piecewise linear function.

```
. mkspline dose 5 = dosage
. logistic outcome dose1-dose5
oom
```

‘`mkspline dose 5 = dosage`’ creates five variables, `dose1`, `dose2`, ..., `dose5`, equally spacing the knots over the range of `dosage`. If dosage varied between 0 and 100, ‘`mkspline dose 5 = dosage`’ has the same effect as typing

```
. mkspline dose1 20 dose2 40 dose3 60 dose4 80 dose5 = dosage
```

The `pctile` option sets the knots to divide the data into 5 equal sample-size groups rather than 5 equal-width ranges. Typing

```
. mkspline dose 5 = dosage, pctile
```

places the knots at the 20th, 40th, 60th, and 80th percentiles of the data.

Methods and Formulas

Let V_i, $i = 1, \ldots, n$ be the variables to be created, k_i, $i = 1, \ldots, n-1$ be the corresponding knots, and $\mathcal{V}$ be the original variable (the command is '`mkspline` V_1 k_1 V_2 k_2 ... V_n `=` $\mathcal{V}$'). Then:

$$V_1 = \min(\mathcal{V}, k_1)$$
$$V_i = \max(\min(\mathcal{V}, k_i), k_{i-1}) - k_{i-1} \quad i = 2, \ldots, n$$

If the `marginal` option is specified, the definitions are

$$V_1 = \mathcal{V}$$
$$V_i = \max(0, \mathcal{V} - k_{i-1}) \quad i = 2, \ldots, n$$

In the second syntax, `mkspline` *stubname # =* $\mathcal{V}$, let m and M be the minimum and maximum of $\mathcal{V}$. Without the `pctile` option, knots are set at $m + (M-m)(i/n)$ for $i = 1, \ldots, n-1$. If `pctile` is specified, knots are set at the $100(i/n)$ percentiles, $i = 1, \ldots, n-1$. Percentiles are calculated by `egen`'s `pctile()` function.

dt1 Five data sets for teaching

J. Theodore Anagnoson, Department of Political Science, California State University, Los Angeles, 213-343-2230

Introduction

Stata is a popular program for classroom use. The program is easy to learn and use, a key consideration in adopting a data analysis package. Just as important, Stata offers state-of-the-art statistical and graphical tools. Many schools already have site licenses for Stata. Students can easily have their own copies of Stata as well. Students are eligible for an academic discount on the full Stata package. Alternatively, they may purchase a student version of Stata for about the price of a textbook. In fact, several recent textbooks—such as Hamilton (1993) and the forthcoming Anagnoson and DeLeon (1994)—come packaged with this student version.

While Stata comes with several data sets that can be used for demonstrations and exercises, instructors often collect their own in order to demonstrate techniques covered in class. This note describes five different data sets I have collected and have found to be useful for class exercises. Hamilton (1993) also includes several data sets on disk, and Hamilton (1992), while not including a disk, has many interesting data sets that easily can be entered into Stata and used for classroom exercises and demonstrations.

The five data sets described below are included on the STB diskette.

Data set 1: Union membership in advanced industrial nations

The first data set, `unionmem.dta`, comes from an article by Stephens in the September 1991 issue of *American Political Science Review*. There are twenty observations corresponding to the twenty OECD nations included. The data set contains information on the size of the labor force, the percent of the work force that is unionized, the Wilensky index of the political leanings of the government, the Atlas-Naradov-Mira index of ethnic diversity, the Muller index of linguistic diversity, and a measure of economic concentration.

Figure 1 displays the relationship between the percent of the workforce that belongs to a union and the political leanings of the government. The relative positions of most of the countries appear to make sense. Ireland is clearly an important outlier though.

Another useful feature of these data for classroom use is the wide variation in country size. This variation in size makes these data useful for demonstrating the use of weights in statistical analysis.

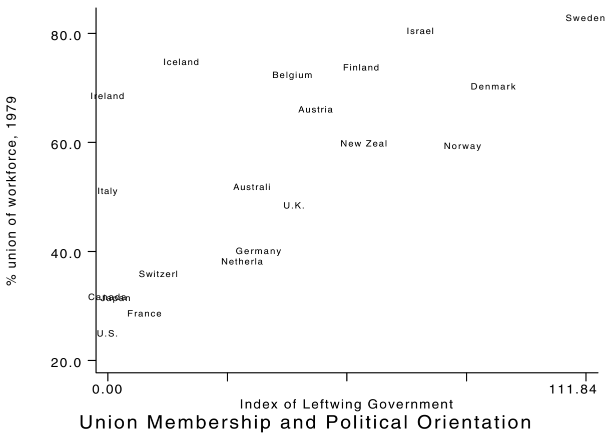


Figure 1: `unionmem.dta`

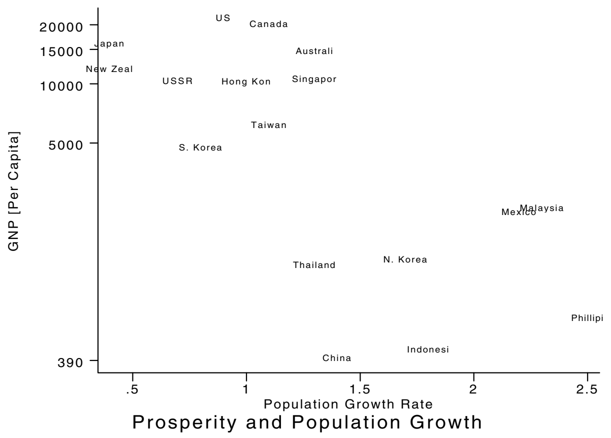


Figure 2: `pacrim.dta`

Data set 2: Pacific Rim nations

The second data set, `pacrim.dta`, appeared in a special "world" section of the *Los Angeles Times* on May 21, 1991 and contains data on various characteristics of the 17 Pacific Rim nations. Again, one of the interesting problems is the range of size of these nations, from China and the U.S. down to New Zealand, Singapore, and Hong Kong. The variables include population, population growth, land area, gross national product per capita, the literacy rate, persons per doctor, infant mortality per 1,000 live births, and life expectancy for males and females.

Figure 2 displays the relationship between the log of gross national product per capita and the population growth rate. Note how the data separate into two clusters. Note also that South Korea is a member of the more prosperous cluster, while North Korea is a member of the less prosperous cluster. The GNP figures for the USSR, China, and North Korea may be highly inaccurate, because of poor data availability and because none of these countries has a convertible currency.

Data set 3: Gulf War nations

Associate editor Richard DeLeon from San Francisco State University compiled this data set from various tables in the United Nations Human Development Report. There is one observation each on six Middle Eastern countries and the United States. The variables in this data set include gross domestic product per capita, life expectancy at birth, the adult literacy rate, military spending as a percent of gross national product, the ratio of military spending to spending on health and education, and spending on arms imports. These data can be used to examine the tradeoff between military spending and other types of social spending.

Figure 3 displays the relationship between the log of gross domestic product per capita and the ratio of military spending to spending on health and education. To my eye, the data form three clusters, although one must be cautious about overinterpreting so few data points. Nonetheless, the U.S. and Kuwait appear to form a high income/low military spending cluster; Saudi Arabia and Israel appear to form a medium income/medium military spending cluster; and Jordan, Iran, and Iraq appear to form a low income/high military spending cluster.

Data set 4: Fertility rates of U.S. women

I compiled this data set from an almanac (Wright 1989). It contains the general and total fertility rates of U.S. women from 1930 through 1988. The general fertility rate is the ratio of live births to the total number of women aged 15 to 44. The total fertility rate is "the number of births 1,000 women aged 10 to 50 would have in their lifetimes if at each year of age they experienced the birth rates occurring to women of that age in the specified calendar year" (Wright 1989, 240). The total fertility rate is sometimes identified in the media as the number of births a women is likely to have in her lifetime. This rate is most helpful in measuring long-term trends, especially in determining whether or not the nation is sustaining a level of reproduction necessary for maintaining current population trends. That rate is usually set at 2,100 children per 1,000 women, a rate that has not been seen in the U.S. since 1971.

Figure 4 shows four views of the total fertility rate along with a resistant nonlinear compound smoothed version of the series. As you might expect, the observed data are least smooth around World War II. The observed total fertility rate exceeds the smoothed values early in the war but drops sharply as the war continues. The observed fertility rate in 1947 is much higher than the smoothed value, because the smoothed series adjusts slowly to the postwar baby boom. These data demonstrate the ability of this type of smoother to highlight unusual data points.

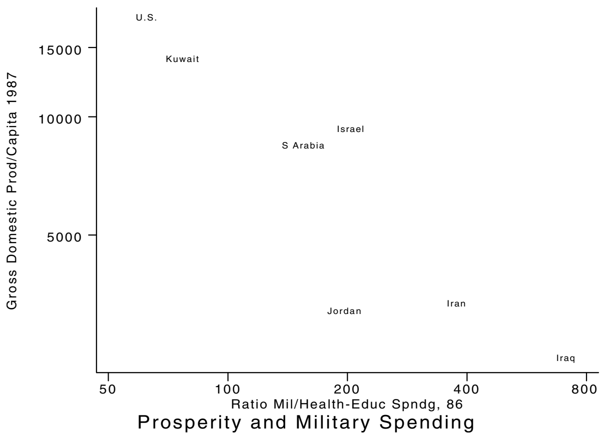


Figure 3: `gulfwar.dta`

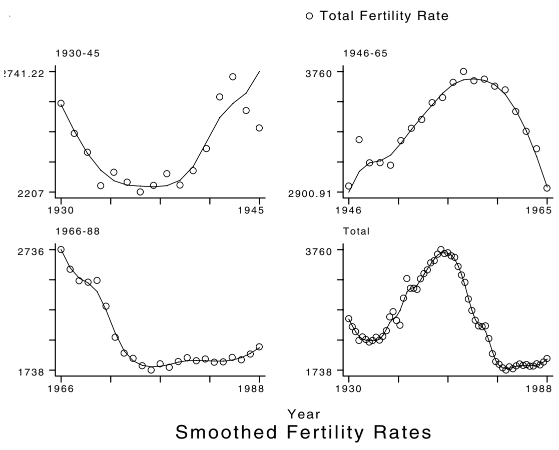


Figure 4: `fertilty.dta`

Data set 5: Executions in California

The final data set contains information on executions in California from 1893 through 1967 along with a dummy variable that is equal to 1 in gubernatorial election years. Figure 5 displays the actual number of executions along with a smoothed version of this series. Election years are highlighted.

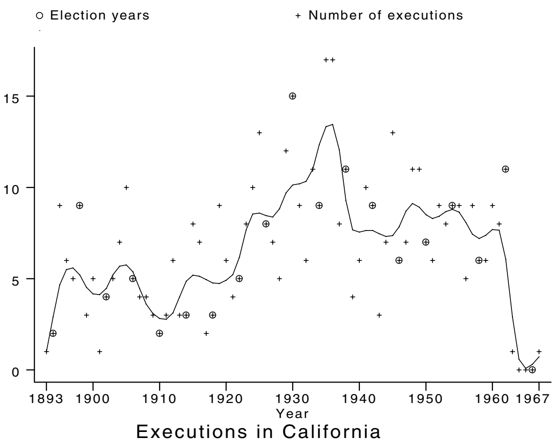


Figure 5: `executed.dta`

References

Anagnoson, J. T. and R. E. DeLeon. 1994. *Using StataQuest, A Student's Guide to Data Analysis Using StataQuest.* Belmont, CA: Wadsworth.

Hamilton, L. C. 1993. *Statistics With Stata.* Belmont, CA: Duxbury.

——. 1992. *Regression With Graphics, A Second Course in Applied Statistics.* Pacific Grove, CA: Brooks/Cole.

Muller, S. H. 1964. *The World's Living Languages: Basic Facts of Their Structure, Kinship, Location and Number of Speakers.* New York: Ungar.

Stephens, J. D. 1991. Industrial concentration, country size, and trade union membership. *American Political Science Review* 85: 941–953.

Taylor, C. L. and M. C. Hudson. 1972. *World Handbook of Social and Political Indicators.* 2d ed. New Haven: Yale University Press.

U.S. Bureau of the Census. 1988. *U.S. Population Estimates and Components of Change, 1970–1987.* Washington, D.C.: Bureau of the Census.

Wallerstein, M. 1989. Union organization in advanced industrial democracies. *American Political Science Review* 83: 481–501.

Wilensky, H. L. 1981. Leftism, Catholicism, and democratic corporatism: The role of political parties in recent welfare state development. In *The Development of Welfare States in Europe and America*, ed. P. Flora and A. J. Hedenheimer. New Brunswick, NJ: Transaction Books.

Wright, J. W. 1989. *The Universal Almanac 1990.* Kansas City: Andrews and McMeel.

gr13 | Incorporating Stata-created PostScript files into TEX/LATEX documents

Teck-Wong Soon and Sutaip L C Saw, Department of Economics and Statistics
National University of Singapore, FAX (011)-65-775-2646, EMAIL ecsstw@nusvm.nus.sg

Introduction

Many Stata users prepare their papers and reports with TEX, a powerful and inexpensive formatting program for technical and mathematical documents. While TEX can format virtually any kind of document, TEX has no standard method for handling graphics, a serious omission for authors of statistical articles. Stata, on the other hand, produces excellent statistical graphics. Thus it is natural for Stata users to want to incorporate their Stata-created graphs into their TEX documents.

This article explains how to incorporate graphs created by Stata into TEX documents. The process is actually relatively easy. The technique described below works for TEX documents that are printed on a PostScript printer. The process of combining TEX and Stata materials is similar for other printers, but a few details have to be changed to match the document to the requirements of the printer. We will point out these details as we go along. Everything in this article works equally well for TEX and LATEX documents. LATEX is a flavor of TEX that hides TEX's complexity by packaging some of TEX's capabilities in easy-to-use commands (called *macros* in TEX.)

There is tremendous interest in the TEX community in incorporating graphics of all kinds (not just Stata graphs) into TEX documents. Recent articles published in the TUGboat (the communication of the TEX Users Group) on the incorporation of graphics into TEX/LATEX documents include Pickrell (1990), Damrau (1992) and Weiss (1992).

The next section gives an overview of the steps used to incorporate a Stata graph into a TEX document. The succeeding section demonstrates a problem that arises when you try to follow these steps. Next we present a simple solution to this problem. Finally we present an example.

Overview

Stata stores its graphs in a special format (as `.gph` files) that neither printers nor other programs understand. However, the print drivers supplied with Stata—`gphdot`, `gphpen`, and their associated `.dot` and `.pen` files—convert `.gph` files to forms that can be printed by or incorporated into other programs. (See, for example, DeLeon 1991, Hilbe 1991, Saving and Montgomery 1992, and Jacobs 1992 for different approaches to incorporating Stata graphs into WordPerfect and other word processing programs.) Since we need to incorporate Stata graphs into PostScript files, we use the `gphpen` program along with the `ps.pen` driver and `ps.plf` printer injection file to convert our `.gph` files to PostScript form. For example, imagine we had created a Stata graph and saved it on disk, using the `saving()` option of the `graph` command, in a file named `bar.gph`. To convert this file to PostScript format, we could `exit` Stata and give the command

```
gphpen bar /dps
```

specifying the PostScript driver (`ps.pen`) with the `/d` option. This example assumes we are working on a DOS system and that the `ps.plf` file is either in the current directory, in the path, or in the `\stata` subdirectory. (If this description is confusing, read [3] printing in the *Stata Reference Manual*—especially the section titled *Creating PostScript files*—for more background.)

When `gphpen` converts a `.gph` file to PostScript format, it saves the PostScript output in a file with the extension `.ps`. (Other drivers may send the output directly to the printer by default.) In this case, `gphpen` automatically creates the file `bar.ps`.

All that's left now is to tell TeX to include the `bar.ps` file at the appropriate place in our document. There is a complication, though. By design, TeX has no mechanism for incorporating graphics. To explain why this is so, and to explain how we can, nonetheless, incorporate our Stata graph into our TeX document, we have to tell you a little bit about how TeX works. If you're already an experienced TeX user, skip ahead four paragraphs (to the paragraph that starts "Graphics formats are also specific to devices"). If you're new to TeX or if you've never really understood what happens when you TeX your document, continue reading.

TeX is a formatting program, or what is called a *markup language*. Authors edit their TeX documents in plain ASCII files (for example, `myfile.tex`). Authors *markup* their text by including TeX or LaTeX commands that describe the appearance of the final document. For instance, to change to a boldface font, the author includes the command `\bf`. Other document features (title pages, section headings, etc.) are indicated by including the appropriate TeX commands.

A markup language like TeX contrasts with what-you-see-is-what-you-get (WYSIWYG) word processing programs such as WordPerfect and Microsoft Word which use proprietary codes to indicate, both in the document and on the screen, changes in appearance such as font changes. The advantage of WYSIWYG is the ease of scanning the final product on screen during editing. The advantage of TeX is that the program can be used on virtually any computer using any monitor. In addition, authors can use the editor of their choice to produce their TeX documents. There are no hidden codes stored in a `.tex` file.

When it is time to print a TeX document, the author runs two programs. First, the author TeX's the document, that is, the author uses TeX to convert the document to a device-independent format that can, again, be stored on any computer. For instance,

```
tex myfile
```

converts `myfile.tex` to `myfile.dvi` where '`.dvi`' stands for 'device-independent' format. Next the author uses a second program called a print driver that converts `.dvi` files to a form acceptable to the author's printer. For instance, the author might type

```
dvips myfile
```

to convert `myfile.dvi` to `myfile.ps`, a PostScript version of the document.

The print driver is not a part of TeX. Remember that TeX is 'device-independent', that is, TeX will run on any computer. However, printers have very specific requirements. For every printer/computer combination, there is a driver that converts the device-independent `.dvi` file to the format required by the printer.

Graphics formats are also specific to devices, that is, graphics must be stored in forms that can be understood and displayed by specific monitors and printers. As a consequence, it is the print driver that incorporates Stata graphs, and other graphics, into TeX documents. Almost all print drivers have some facility for incorporating graphics, but each print driver has its own conventions that the author must obey, if the process is to be successful.

Although TeX has no built-in commands to handle graphics, Donald Knuth, the author of TeX, anticipated this need and added a command, called `\special`, to handle this problem. Anything can be typed in a `\special` command; TeX ignores it. The `\special` command just provides a way to pass instructions to the print driver. For example, the command

```
\special{ps:bar.ps}
```

might tell the print driver to include the PostScript file `bar.ps`.

In fact, this *is* the way to tell one particular print driver to include `bar.ps`. Remember, though, that each print driver has its own conventions for `\special` commands. You have to read the manual for your print driver to find out the form of the `\special` command you need to use. Alternatively, you can get a copy of the PostScript driver written by Tomas Rokicki of Stanford University. This driver comes with specially-written TeX macros that will handle these details for you. This driver is available through the Internet via anonymous FTP from almost all the standard TeX repositories, or you can contact Rokicki by EMAIL at `rokicki@cs.stanford.edu`

There is one last detail you must take account of when incorporating your Stata graph. As we said, TeX ignores the `\special` command. In particular, TeX has no way of knowing a graph has been included or how large the graph is. In order to prevent the graph from overprinting the text of the document, you have to instruct TeX to, in essence, 'move the cursor' to a point on the page just beyond the graph. There are three ways to do this. First, you can use some simple TeX commands to move the cursor where you want it. Second, many print drivers save you this trouble by including TeX macros that combine these commands with the appropriate `\special` command; check your manual. Finally, the macros that come with the Rokicki driver will also handle these details for you.

This overview has covered a lot of ground. It may be useful at this point to summarize what we've said so far. To incorporate a Stata graph into a TeX document, you must

1. Create and save a Stata graph, e.g.:

   graph ..., ... **saving(bar)**,

2. Use **gphdot** or **gphpen** to convert your **.gph** file to a format your printer can understand, e.g.:

   gphpen bar /dps,

3. Add the appropriate TEX commands to your document to incorporate the graph. For example, you can add TEX commands to move the 'cursor' to the desired location, then add the **\special** command your print driver requires. If you use Rokicki's macros and driver, you might add the following code to your LATEX file:

```
\begin{center}
\leavemode
\epsfxsize=3.5in
\epsfbox{bar.ps}
\end{center}
```

The first, second, and fifth lines are ordinary LATEX commands. The third and fourth lines are macros supplied by Rokicki. The third line defines the width of the graph, while the fourth line gives the name of the file in which the graph is stored,

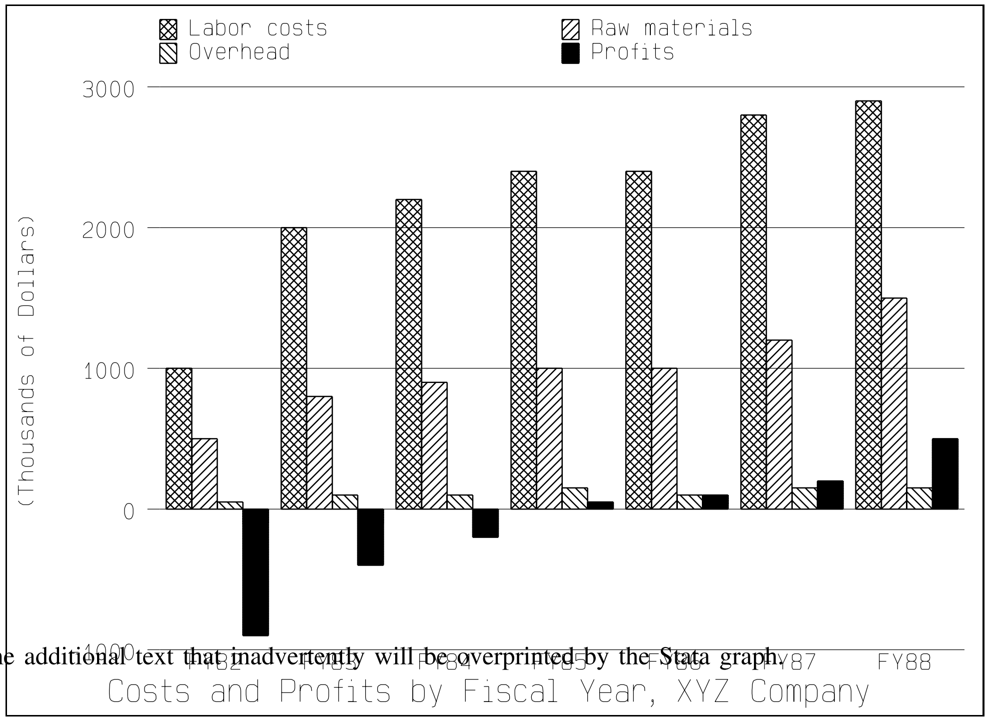
4. Finally, TEX (or LATEX) the document, then use the print driver to print it.

As you can see, the process is really quite simple, even though it took us quite a while to describe it.

The problem

Unfortunately, there is still a problem with the procedure we outlined above. The PostScript file produced by **gphpen** does not 'mesh' correctly with the TEX macros. Here is a demonstration of the problem. The bar graph below was produced by placing the following lines in our LATEX document:

```
\begin{center}
\leavemode
\epsfxsize=3.5in
\framebox{\epsfbox{bar.ps}}
\end{center}
Here is some additional text that inadvertently will be
overprinted by the Stata graph.
```



Here is some additional text that inadvertently will be overprinted by the Stata graph.

The LaTeX `\framebox` command outlines the area where Stata 'told' TeX to expect the graph to appear. As you can see, Stata misled TeX; the graph is much larger than Stata claimed it was.

The cause

The problem lies in the file `ps.plf` which specifies how PostScript is to interpret the Stata graphics file. When `gphpen` creates a PostScript version of a `.gph` file, it includes the contents of `ps.plf` at the beginning of the file. (`.plf` stands for *P*re*L*oaded *F*ile.) Among other things, `ps.plf` defines the *bounding box*, the construct PostScript uses to define the size of a graphics image.

The bounding box, in effect, bounds the image. If the bounding box is defined correctly, other information—in this case, the text of the TeX document—can be placed anywhere outside the bounding box without danger of overprinting the graph. Unfortunately, `ps.plf` defines the bounding box incorrectly. When `gphpen` is used to print a Stata graph, the graph is the only thing on the page, and the incorrect bounding box information doesn't cause a problem. When the `.ps` file is incorporated into a TeX document, however, the incorrect bounding box information confuses TeX and leads to the results you see in the example above.

The obvious solution

The obvious solution to the problem is to determine the appropriate bounding box. Specifying the correct bounding box will convert the PS file into a well-specified EPS (Encapsulated PostScript) file which can be incorporated easily into our document with `epsf.tex`/`epsf.sty` and `dvips`.

Fortunately, PostScript files are stored as plain ASCII files, that is, they can be read and edited like any other ASCII file. This means that we can correct the bounding box instruction in the `.ps` files produced by `gphpen`.

The first two lines of the file "bar.ps" are shown below:

```
\begin{center}
%!PS-Adobe-2.0 EPSF-1.2
%%BoundingBox:  0 0 396 504
```

Every PostScript file begins with two statements like these. The first line gives information about the version of the PostScript page description language used in the file. The second line describes the bounding box. We need to modify the second line to specify an appropriate bounding box.

After close examination of the printed graph, we determined that the appropriate bounding box is `0 0 458 396`. We used our editor to change `bar.ps` so the first two lines read:

```
\begin{center}
%!PS-Adobe-2.0 EPSF-1.2
%%BoundingBox:  0 0 458 396
```

With this minor modification, the PostScript graphic image now is placed in our TeX document in the desired position, properly scaled, and with the correct amount of space provided in the document for surrounding text. The effect of this modification is to change the upper right corner of the bounding box to reflect the boundary of the actual graphic image.

Refinement

The obvious solution described above fixes the problem, but it is too tedious. It's just not convenient to manually edit every PostScript file generated by `gphpen`. A better solution is to modify `ps.pen` and `ps.plf` so `gphpen` automatically produces PostScript files with correct bounding boxes. We made the needed modifications and stored the corrected files under the names `eps.pen` and `eps.plf`. [*These files are available on this issue's* STB *disk—Ed*.] We suggest that the files `ps.pen` and `ps.plf` be retained, as they are appropriate if we wish to print stand-alone Stata graphs in PostScript.

We modified lines 10 and 11 of `ps.pen` which read:

```
ps                  * Default output filename extension
´#include ps´       * Initialization file for device
```

We modified these two lines to read:

```
eps                  * Default output filename extension
´#include eps´       * Initialization file for device
```

Line 10 specifies the file extension for the PostScript file produced by `gphpen`. We changed that extension to `.eps` to avoid confusion with files produced by `ps.pen` and `ps.plf`. Line 11 specifies the filename of the `.plf` to include at the top of the PostScript file. `eps.pen` includes `eps.plf`, our version of this header file, rather than `ps.plf`, the file provided with Stata.

The substantive changes are in `eps.plf` which contains the PostScript commands that control the appearance of the printed graph. The first two lines of `ps.plf` are the same as the first two lines of the graph file (`bar.ps` in our example), and the changes are the same ones described in the previous section, that is, the numbers in line 2, the bounding box line, are changed from 0 0 396 504 to 0 0 458 396.

We made another important change to `ps.plf`. The last few lines of the original file read:

```
initgraphics
.072 dup 680 4760 ginit

% Following is STATA-generated page description:
```

We changed these lines to read:

```
%initgraphics
.072 dup 0 0 ginit

% Following is STATA-generated page description:
```

The percent sign (%) is the comment character in PostScript (and in TeX), so our first modification comments out the `initgraphics` command. The second modification changes the origin of the graph in the PostScript coordinate system. After some experimentation, we found that these changes were necessary to permit the TeX macros in `epsf.sty` to successfully rescale the Stata graph.

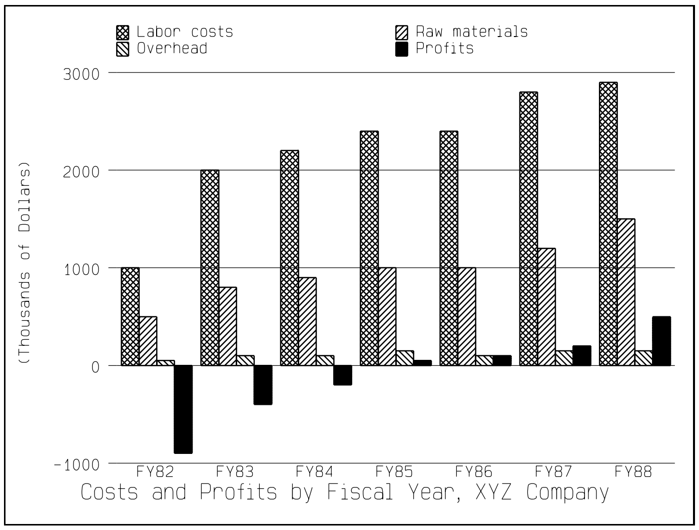
Result

The problem is now solved, once and for all. We use our modified PostScript driver files to produce a corrected Stata graph as follows:

```
gphpen bar /deps
```

Now for the acid test. Our LaTeX document reads:

```
\begin{center}
\leavemode
\epsfxsize=3.5in
\framebox{\epsfbox{bar.ps}}
\end{center}
Here is some additional text that is {\em no longer}
overprinted by the Stata graph.
```



Here is some additional text that is *no longer* overprinted by the Stata graph.

If you compare the image above to the first example in the insert, you'll notice that the graph not only is located correctly on the page, it also is resized (by the `eps`f*xxx* TEX macros) to fit into the LATEX framebox. We invite you to try your own experiments to see just how easy it is to place your Stata graphs wherever you wish in your TEX documents. If you begin to do this regularly, you may wish to copy `eps.pen` to `default.pen` to save typing the `\d` device driver option in your `gphpen` command.

References

Damrau, J. 1992. Discovering graphics in LATEX documents. *TUGboat* 13: 315–321.

DeLeon, R. E. 1991. Importing Stata's graphs into MS-Word or WordPerfect. *Stata Technical Bulletin* 2: 5–6.

Hilbe, J. 1991. Using Stata Graphs in the Windows 3.0 Environment. *Stata Technical Bulletin* 3: 9–10.

Jacobs, M. 1992. Printing graphs and creating WordPerfect graph files. *Stata Technical Bulletin* 7: 5.

Knuth, D. E. 1986. *The TEXbook*. Reading, MA: Addison–Wesley.

Lamport, L. 1986. *LATEX: A Document Preparation System*. Reading, MA: Addison–Wesley.

Pickrell, L. S. 1990. Combining graphics with TEX on IBM PC-compatible systems and LaserJet printers. *TUGboat* 11: 26–31.

Saving, T. R. and J. Montgomery. 1992. Printing graphs and creating WordPerfect graph files. *Stata Technical Bulletin* 5: 6–7.

Weiss, N. A. 1992. Creation and incorporation of PostScript graphics with TEX-formatted labels into TEX documents. *TUGboat* 13: 330–334.

gr13.1	`\special{}` effects with Stata graphs in TEX documents

Sean Becketti, Editor, STB, FAX 913-888-6708

The previous insert, "Incorporating Stata-created PostScript files into TEX/LATEX documents" by Teck-Wong Soon and Sutaip L C Saw, provides step-by-step instructions for combining Stata graphs and TEX documents and corrects a problem with Stata's PostScript driver. Astute readers, however, may have noticed that the STB is printed using TEX and that Stata graphs regularly appear in the pages of the STB. Why then, you may ask, has the problem with PostScript driver not been corrected or discussed in previous STBs?

The answer is simple: the STB is not printed on a PostScript printer. The STB is printed on a Hewlett-Packard LaserJet using PCL. As a consequence, we incorporate Stata graphs into the STB using the `\special` command supported by the print driver that converts our `.dvi` files to HP PCL format. In fact, we've written several macros of our own to make it easier to place Stata graphs in STB inserts. Thus, we simply never encountered the problem described in *gr13*.

This explanation raises another question, though. How did we print the PostScript graphs Soon and Saw presented in *gr13*? Again, the answer is simple: we didn't. Instead, we used our HP driver's `\special` commands and some TeX tricks of our own to reproduce the appearance of the PostScript graphs submitted by Soon and Saw. (We did, however, test Soon and Saw's files on a PostScript printer, and they did work.)

We are trying to make a couple of points here. First, incorporating Stata graphs into TeX documents is fairly easy. We use and endorse the general approach described in *gr13* by Soon and Saw, although there are other ways of accomplishing the same result. And that is our second point: there is more than one approach for incorporating Stata graphs into other documents. In the case of *gr13*, we needed to imitate a PostScript problem in a LaTeX document using an HP PCL printer and plain TeX.

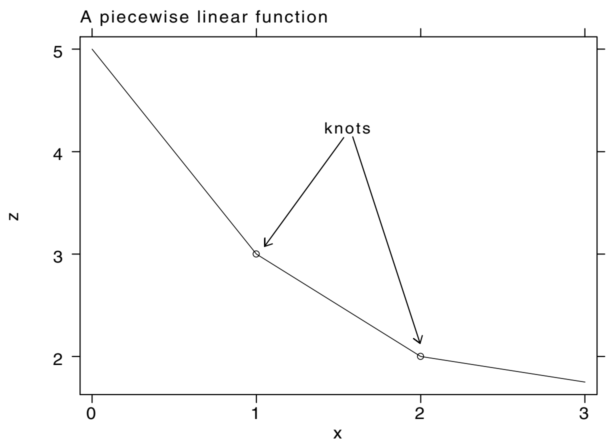
We encourage you to develop your own methods for incorporating Stata graphs into your documents. The articles cited at the end of *gr13* give some tips for incorporating Stata graphs into WordPerfect and Microsoft Word documents. Other document preparation systems can be handled as well. Be sure to let us in on any tricks or improved techniques you discover so we can share them with other STB readers.

sg19 Linear splines and piecewise linear functions

William Gould, Stata Corporation, FAX 409-696-4601

[Also see crc33 in this issue for a related insert.—Ed.]

Linear splines allow estimating the relationship between y and x as a piecewise linear function. A piecewise linear function is just that: it is a function composed of linear segments—straight lines. One linear segment represents the function for values of x below x_0. Another linear segment handles values between x_0 and x_1, and so on. The linear segments are arranged so that they join at x_0, x_1, ..., which are called the knots. An example of a piecewise linear function is shown below.



Piecewise linear functions can be used to approximate true nonlinear relationships in data. They have the advantage that the shape is data driven; it will not be an artifact of functional-form assumptions. For instance, using the automobile data, let us attempt to model the relationship between price and mileage rating. We begin with the assumption:

$$\texttt{price}_j = f(\texttt{mpg}_j) + u_j$$

We are going to use this simple model because it allow us to draw graphs to compare statistical results. I ask, however, that you pretend our model is

$$\texttt{price} = \mathcal{B}'\mathbf{x}_j + f(\texttt{mpg}_j) + u_j$$

In this model, determining the functional form of $f()$ is not as easy because one cannot draw simple graphs. Moreover, I ask that you also pretend that your interest in is $\mathcal{B}$, not in $f()$ itself. I ask that only because, were that really the case, you might be tempted to use the following popular "solution" to parameterizing the nuisance parameter mpg:

$$\texttt{price} = \mathcal{B}'\mathbf{x}_j + \gamma\,\texttt{mpg}_j + u_j$$

After estimating such a model, many researchers would claim that they have "controlled for" mpg. In my simple model, this is equivalent to:

$$\texttt{price} = \alpha + \gamma\,\texttt{mpg}_j + u_j$$

The regression results are

```
. regress price mpg
  Source |       SS       df       MS                  Number of obs =      74
---------+------------------------------               F(  1,    72) =   19.26
   Model |  134033225     1   134033225                Prob > F      =  0.0000
Residual |  501032171    72  6958780.16                R-square      =  0.2111
---------+------------------------------               Adj R-square  =  0.2001
   Total |  635065396    73  8699525.97                Root MSE      =  2637.9

------------------------------------------------------------------------------
   price |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |  -223.7948   50.99297     -4.389   0.000      -325.4474   -122.1422
   _cons |   10961.72    1135.11      9.657   0.000       8698.923    13224.53
------------------------------------------------------------------------------
```

This is a reasonable looking regression, but it is not so reasonable when we look at a graph of the data and the line we have just fitted, as shown in Figure 1. (In the more complicated model, we could not look at this simple graph and so might not discover the unreasonableness of our assumption.)

A more careful researcher might worry about the linearity assumption at the outset. Linearity means that the effect on price of a change in mpg is the same regardless of the value of mpg. Some things operate this way, but many do not. Such a careful researcher might estimate a quadratic:

$$\texttt{price} = \mathcal{B}'\mathbf{x}_j + \gamma_0\,\texttt{mpg}_j + \gamma_1\,\texttt{mpg}_j^2 + u_j$$

Estimating such a model in my simple case results in

```
. gen mpgsq = mpg^2
. regress price mpg mpg2
  Source |       SS       df       MS                  Number of obs =      74
---------+------------------------------               F(  2,    71) =   18.26
   Model |  215699821     2   107849910                Prob > F      =  0.0000
Residual |  419365576    71  5906557.40                R-square      =  0.3396
---------+------------------------------               Adj R-square  =  0.3210
   Total |  635065396    73  8699525.97                Root MSE      =  2430.3

------------------------------------------------------------------------------
   price |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |  -1247.567    279.306     -4.467   0.000      -1804.487   -690.6464
   mpgsq |   20.86536   5.611395      3.718   0.000       9.676555    32.05416
   _cons |   22564.58   3290.976      6.857   0.000       16002.56     29126.6
------------------------------------------------------------------------------
```

This model also appears most reasonable and, were we the careful researcher, we could take satisfaction in having fit a better model than the researcher who controlled for the effect of mpg by merely including a linear term. Unfortunately, our satisfaction dwindles when, in this simple model, we look at the graph shown in Figure 2 (a graph that could not be easily drawn in the more complicated model). The graph exhibits a classic problem with the quadratic—the direction of the effect eventually reverses itself. Although the quadratic is widely used to approximate nonlinear functions, it is typically done under the assumption that the turning point lies outside the data.

Even if one cannot look at the graph, one can calculate the point at which the effect changes direction. Given that we parameterize an effect as $\gamma_0 z + \gamma_1 z^2$, the reversal occurs at $-\gamma_0/(2\gamma_1)$. Whenever one estimates using a quadratic, one should make this calculation. Substituting the values we have estimated, we obtain

$$\frac{1247.567}{2 \times 20.86536} = 29.90$$

The range of mpg in our data is 12 to 41, so the reversal does occur within the range of our data—just as the graph showed, but we are pretending that we could not look at the graph. In any case, we must now ask ourselves whether the reversal is real—whether price really does decrease with mpg but eventually turns around and actually increases, or whether the upturn is an artifact of our quadratic assumption. In the case of the simple model, we can look at a graph and we do indeed see some, but not overwhelming, evidence that the upturn is real. In a more complicated model, we could not look.

We could, however, attempt to fit the function with a piecewise linear function. We could do this at the outset or, having fit a quadratic and having observed the reversal point inside the range of our data, do so afterwards to make sure that the upturn in real. In testing the quadratic, it is important that we set one of the knots to the point at which the effect changed directions. If the effect is real, the piecewise linear function will presumably change directions just as the quadratic did. The mkspline command (see *crc33* in this issue) makes construction of spline functions easy:

```
. mkspline mpg1 20 mpg2 30 mpg3 = mpg
. regress price mpg1 mpg2 mpg3
  Source |       SS       df       MS                  Number of obs =      74
---------+------------------------------               F(  3,    70) =   14.62
   Model |  244597553     3  81532517.7                Prob > F      =  0.0000
Residual |  390467843    70  5578112.04                R-square      =  0.3852
---------+------------------------------               Adj R-square  =  0.3588
   Total |  635065396    73  8699525.97                Root MSE      =  2361.8
------------------------------------------------------------------------------
   price |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    mpg1 |   -817.042   148.9579     -5.485   0.000      -1114.129   -519.9548
    mpg2 |  -11.17754   112.0447     -0.100   0.921      -234.6436    212.2886
    mpg3 |  -14.01031   187.0467     -0.075   0.941      -387.0631    359.0425
   _cons |   21260.16   2633.778      8.072   0.000       16007.26    26513.07
------------------------------------------------------------------------------
```

Remember, the reversal point—the point at which the quadratic parameterization predicted price started increasing with mileage rating—was 29.90. mpg3 in the regression above corresponds to the mpg range 30 and above. In the piecewise-linear specification, the effect is estimated as being negative, although the confidence interval is, admittedly, quite wide. The effect might be positive and, in fact, it might be positive going all the way back to 20 mpg, but we cannot reject that it is negative or zero. Figure 3, a luxury we would not have in a more complicated model, reveals that the piecewise-linear fit is quite reasonable.

Other alternatives to the piecewise-linear fit

A favorite alternative to quadratics and piecewise linear functions is to categorize the data. The logic goes like this: I do not know the relationship between price and mileage, therefore I will place mpg into categories and use dummy variables to represent each category. The logic is the same as the logic justifying the piecewise-linear fit and as a matter of fact, the categorization method suffers from all the same problems as the piecewise-linear approach: Selecting the points at which an observation shifts from one category to the next has the same arbitrariness as selecting the knots for the linear spline. In any case, we can fit our data with the categories:

```
. gen mpgcat = recode(mpg,20,30,40)
. quietly tab mpgcat, gen(m)
. regress price m2 m3
  Source |       SS       df       MS                  Number of obs =      74
---------+------------------------------               F(  2,    71) =    3.21
   Model | 52605326.5     2  26302663.3                Prob > F      =  0.0464
Residual |  582460070    71  8203662.95                R-square      =  0.0828
---------+------------------------------               Adj R-square  =  0.0570
   Total |  635065396    73  8699525.97                Root MSE      =  2864.2
------------------------------------------------------------------------------
   price |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      m2 |  -1403.749   699.5293     -2.007   0.049      -2798.571   -8.927297
      m3 |  -2503.316   1258.238     -1.990   0.050      -5012.171    5.539538
   _cons |   6937.316   464.6352     14.931   0.000        6010.86    7863.772
------------------------------------------------------------------------------
```

Once again, we see a wide confidence interval but no strong evidence of an upturn; Figure 4 graphs the result.

This approach is perhaps better than the linear approach or quadratic approach (it is arguable), but I would argue that it is inferior to the piecewise-linear approach. First, the graphs reveal that the piecewise linear function fits the data better (as do the R^2's: the piecewise-linear R^2 is .3852 while the dummy R^2 is only .0828). Second, the model—taken literally—has a silly implication. Mileage rating does not affect price until it changes from $20-\epsilon$ to $20+\epsilon$ or from $30-\epsilon$ to $30+\epsilon$, at which point it takes a discrete jump. One is forced into such models when all one has is categorized data, but it would be a shame to throw away the observed variation in mileage rating and substitute categories when we do not have to. The graphs also make clear that the piecewise-linear and dummy parameterizations share much in common: the only difference is that a slope joining the lines is substituted for the jumps between categories. This is a far more reasonable assumption. When one has continuous data, I can think of no reason to substitute categorization for piecewise linearity except that categorization is typically easier. `mkspline` does away with that argument.

Another approach for testing the linearity (or quadratic or any parametric) assumption is to combine the assumption with categorization dummies. At this point, one is not seriously putting forward the model, one is using the model to test that the assumptions are reasonable. In the case of linearity, we can test by including the linear term and our dummies. If the linear assumption is correct, the dummies should be zero.

```
. regress price mpg m2 m3
  Source |       SS       df       MS                  Number of obs =      74
---------+------------------------------               F(  3,    70) =   11.35
   Model |  207797766     3  69265922.0                Prob > F      =  0.0000
Residual |  427267630    70  6103823.29                R-square      =  0.3272
---------+------------------------------               Adj R-square  =  0.2984
   Total |  635065396    73  8699525.97                Root MSE      =  2470.6

------------------------------------------------------------------------------
   price |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |  -583.9617   115.8111     -5.042   0.000      -814.9396   -352.9838
      m2 |   2870.441   1040.484      2.759   0.007         795.26    4945.622
      m3 |   8428.038   2424.403      3.476   0.001       3592.719    13263.36
   _cons |   16833.93   2003.195      8.404   0.000       12838.68    20829.18
------------------------------------------------------------------------------

. test m2 m3
 ( 1)  m2 = 0.0
 ( 2)  m3 = 0.0

       F(  2,    70) =    6.04
            Prob > F =    0.0038
```

The dummies are not zero—we can reject the linear assumption. Figure 5 shows the combined linear plus categorization model. It is obviously ridiculous, but we are not seriously putting this model forth. It is the fact that it is ridiculous—that the coefficients on the dummies are both practically and statistically significant, that leads us to reject the linear model and continue the search for a more reasonable specification.

Piecewise linear functions are one way to find that more reasonable specification.

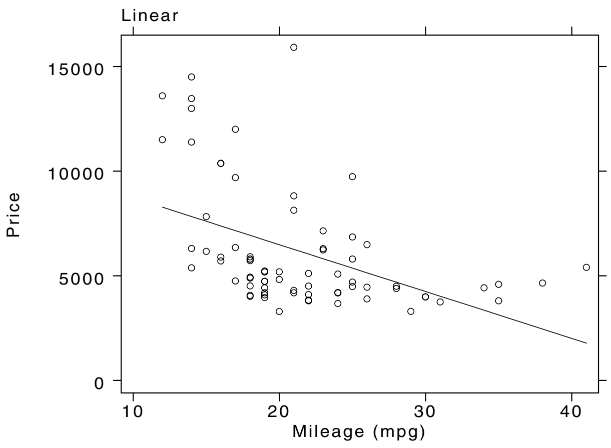


Figure 1

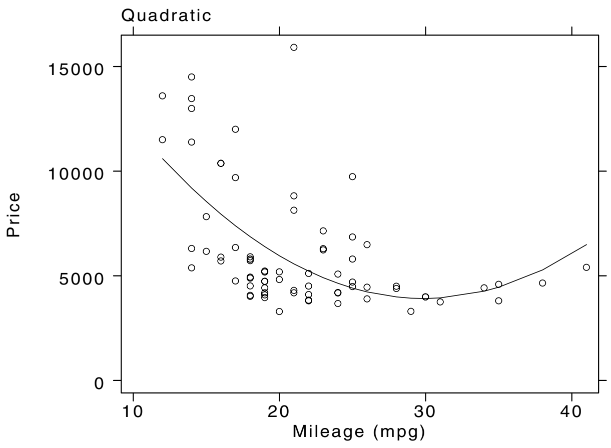


Figure 2

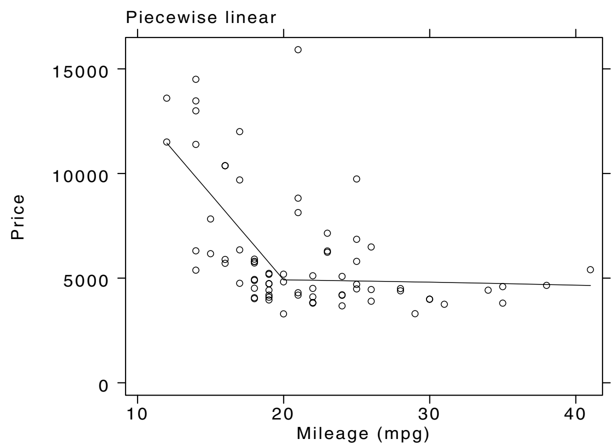


Figure 3

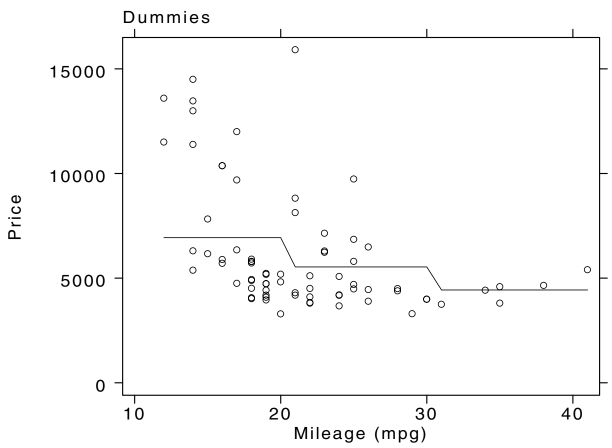


Figure 4

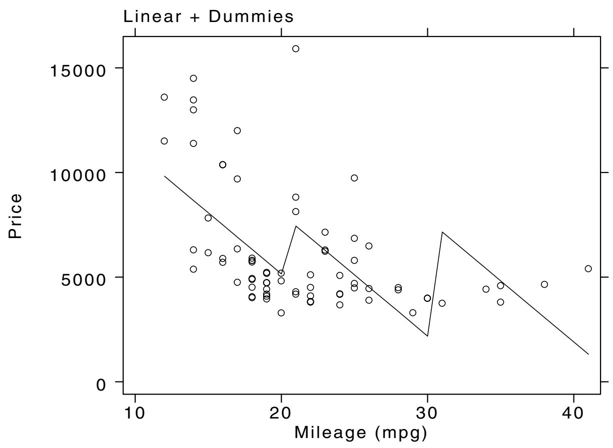


Figure 5

sss1	Calculating U.S. marginal income tax rates

Timothy J. Schmidt, Federal Reserve Bank of Kansas City, 816-881-2307

`mtr`—an addition to the `egen` command ([5d] `egen`)—finds the marginal income tax rate corresponding to any given level of taxable income for a married couple between the years 1930 and 1990. The user typically will have a data set in memory that contains yearly observations on the year and taxable income. When calling the program, the user provides those two series names as arguments to the `mtr` function and specifies a name for the new series that will store the marginal income tax rate data.

The syntax is

egen [*type*] *varname* = **mtr(***year***,***income***)** [**if** *exp*] [**in** *range*]

Alternatively, either *year* or *income* can be replaced with a constant. For example,

```
egen taxrate = mtr(1967,income)
```

Discussion

Many economists have endeavored to study the effects of government fiscal policy on variables like output, employment and prices. Such studies have typically included various measures of government expenditure as their representative fiscal policy variable. However, far fewer researchers have incorporated the tax side of fiscal policy into their models. This rather important omission can be attributed primarily to the inherent difficulty in obtaining data on marginal tax rates. For each level of income in each year, one must find the appropriate rate from a (usually) large schedule in the IRS 1040 tax forms. As a result, papers that have included tax variables at all have often used only summary or aggregate measures such as an average marginal tax rate (e.g. Barro and Sahasakul 1986). However, since the marginal tax rate schedule has changed significantly over time, the use of an average marginal tax rate neglects important information.

The set of programs described herein was written as part of a research paper (Hakkio, Rush, and Schmidt 1993) intended to redress the current lack of a comprehensive data set on marginal income tax rates and thereby promote their use in economic research. The programs were written originally in GAUSS. These programs took as input a time series data set that included the year, various household data, personal income, and total employment. The programs would then generate a distribution of income from that information and compute taxable income and the corresponding time series of marginal income tax rates from the 1040 schedules.

The Stata `mtr` function calculates the marginal tax rate. I have not yet written Stata programs to calculate the family income distribution, taxable income, or the earned income credit. For most applications, `mtr` is all that is needed. If there is enough interest among users, though, I will consider adding these other features to Stata and including them in future STB inserts.

Example

The following example illustrates how income has faced a varying degree of taxation over time. This example uses a hypothetical income distribution generated from the gamma distribution. (These calculations are explained in Hakkio, Rush, and Schmidt 1993.) The data set contains time series representing the 25th, 50th, and 75th percentiles of taxable income from 1930 to 1990. Figure 1 shows how these percentiles of income have grown over this period.

```
. use income
(Income distribution, 1930-1990)
. describe
Contains data from income.dta
  Obs:    61 (max=  5117)                 Income distribution, 1930-1990
 Vars:     4 (max=    99)
Width:    16 (max=   200)
  1. year        float  %9.0g             Year
  2. inc25       float  %9.0g             25th percentile
  3. inc50       float  %9.0g             50th percentile
  4. inc75       float  %9.0g             75th percentile
Sorted by:
. summarize
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
    year |      61        1960   17.75293       1930       1990
   inc25 |      61    6948.272   6891.921      819.4   27227.35
   inc50 |      61    11645.15   11550.71    1373.29   45632.43
   inc75 |      61    18166.43    18019.1    2142.33   71186.58
```

Given these series for income and the (four digit) year, we can generate time series of marginal tax rates corresponding to the percentiles of income.

```
. egen mtr25 = mtr(year,inc25)
. egen mtr50 = mtr(year,inc50)
. egen mtr75 = mtr(year,inc75)
. describe
Contains data from income.dta
  Obs:    61 (max=  5084)                 Income distribution, 1930-1990
 Vars:     7 (max=    99)
Width:    28 (max=   200)
  1. year        float  %9.0g             Year
  2. inc25       float  %9.0g             25th percentile
  3. inc50       float  %9.0g             50th percentile
  4. inc75       float  %9.0g             75th percentile
  5. mtr25       float  %9.0g             25th percentile
  6. mtr50       float  %9.0g             50th percentile
  7. mtr75       float  %9.0g             75th percentile
Sorted by:
Note:  Data has changed since last save
```

```
. summarize mtr25 mtr50 mtr75

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
   mtr25 |      61    .1664898   .0677949     .01125        .23
   mtr50 |      61    .2027508   .0864313     .01125        .29
   mtr75 |      61    .2442053   .1148427     .01125    .424625
```

Figure 2 displays the marginal tax rates corresponding to the 25th, 50th, and 75th percentiles of taxable family income. For the first eleven years of the income tax, these income levels all faced the same marginal tax rate. In recent years, federal income taxes have become more progressive, that is, families with higher incomes have faced higher marginal tax rates. One measure of the progressivity of income taxes is the 'interquartile range', the difference between the marginal tax rate on the 75th percentile of family income and the marginal tax rate on the 25th percentile of family income. Figure 3 displays this measure of progressivity.

References

Barro, R. J. and C. Sahasakul. 1986. Average marginal tax rates from social security and the individual income tax. *Journal of Business* 59: 555–566.

GAUSS software. Maple Valley, WA: Aptech Systems, Inc. (Tel. 206-432-7855)

Hakkio, C. S., M. Rush, and T. J. Schmidt. 1993. The marginal income tax rate schedule from 1930 to 1990. Research Working Paper, Federal Reserve Bank of Kansas City.

Figures

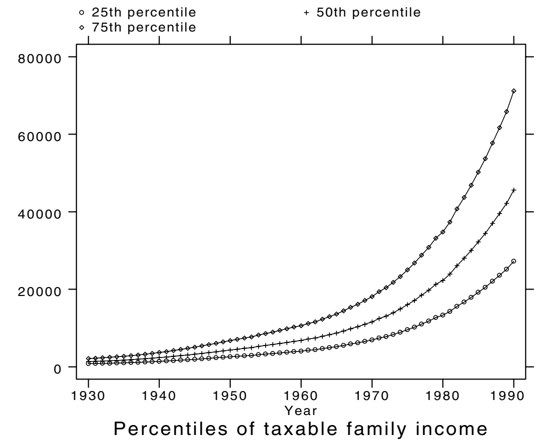


Figure 1

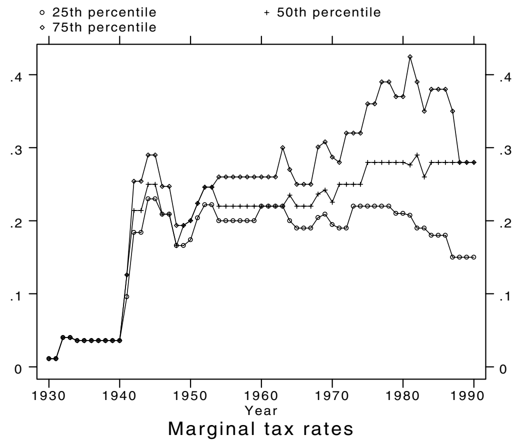


Figure 2

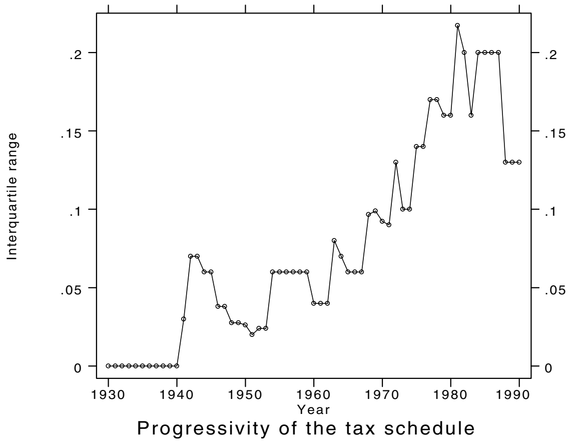


Figure 3

sts4 A suite of programs for time series regression

Sean Becketti, Editor, STB, FAX 913-888-6708

(*The commands presented in this insert are for use with Stata 3.1; see an36.—Ed.*)

Introduction

Time series regressions appear frequently in my research. Many of my colleagues use programs such as PC-GIVE, RATS, or TSP to run these types of regressions. These programs were designed expressly for time series analysis, and they contain many special functions to take account of temporal relationships.

Stata is not specialized for time series analysis. Nonetheless, I use Stata to estimate time series regressions. I find that Stata's flexibility and extensibility outweigh the lack of built-in time series functions. Stata's lack of time series functions is an obstacle though. A good comparison is the relationship between regression and ANOVA. Stata's `regress` command is designed to handle regression, but it can also estimate ANOVA models. One can, with effort, construct the appropriate dummy variables, then use `regress` to calculate least-squares estimates of the primitive ANOVA parameters. Additional tedious calculations can handle tests of the various effects in an ANOVA model. All of this is possible, but it's simply too much work. For this reason, Stata offers a separate `anova` command that automates these calculations and presents the estimates in a form better suited to ANOVA models.

The situation is much the same for time series regression. Stata's `regress` command can estimate these regressions as easily as any others. One can, again with some effort, `generate` the desired lags of the various time series in the regression, and one can use the `test` command to conduct, for example, Granger tests of causality. With a great deal of typing, one can calculate the long-run multipliers implied by the regression. But again, all this is just too much work.

The suite of programs described in this insert automate these aspects of time series regression. These programs create the lags needed in the regressions; estimate the equations; calculate long-run multipliers, permanent effects, and Granger tests; and report a host of diagnostic statistics and model selection criteria.

These programs are the third generation of time series regression routines I have written. The first generation consisted of some crude `do`-files that handled specific equations in which I was interested. The second generation represented the evolution of these `do`-files into full-fledged Stata programs. This second generation of programs was distributed among my colleagues and field-tested for a year. The third-generation programs described in this insert represent a major revision of the second generation programs that incorporates the last year's worth of experience. At my research institution, these programs are used literally every day, and many colleagues who previously used a specialized time series package have switched to Stata and these programs.

tsfit: Estimating a time series regression

The workhorse of time series analysis is ordinary least squares regression. More specialized techniques are sometimes used, but most time series analyses rely almost exclusively on regression.

The complicating factor in time series regression is creating and including lags of the dependent and explanatory variables. A time series regression can be written as

$$A(L)y_t = \gamma_0 + B_1(L)x_{1t} + \cdots + B_J(L)x_{Jt} + \gamma_1 z_{1t} + \cdots + \gamma_K z_{Kt} + \epsilon_t$$

where y_t is the dependent variable, the x's are time series explanatory variables, the z's are non-time series explanatory variables, and ϵ_t is the regression disturbance. $A(L)$, $B_1(L), \ldots, B_J(L)$, and $\gamma_0, \ldots, \gamma_K$ are the regression coefficients.

The letter L in the equation above is the lag operator, that is, L indicates the operation of lagging a variable by one time period. More formally,

$$Lx_t \equiv x_{t-1}.$$

The lag operator can be applied more than once to produce longer lags, thus

$$L^2 x_t \equiv L(Lx_t) = x_{t-2}$$

and

$$L^k x_t \equiv L^{k-1}(Lx_t) = x_{t-k}.$$

The functions of L in the time series regression are lag polynomials. For example,

$$A(L) = 1 - \alpha_1 L - \alpha_2 L^2 - \cdots - \alpha_p L^p$$

and

$$B_1(L) = \beta_{10} + \beta_{11} L + \beta_{12} L^2 + \cdots + \beta_{1q} L^q.$$

By convention, the lag coefficients in $A(L)$ are preceded by minus signs, since they will be moved to the right-hand-side of the equation before the regression is estimated.

From the discussion above, it is clear that a time series regression is defined by the list of time series and non-time series variables included and by the lengths (maximum lags) of each of the lag polynomials. `tsfit` provides a convenient way to specify and estimate time series regressions. The syntax of `tsfit` is

`tsfit` [*varlist*] [`if` *exp*] [`in` *range*] [*weight*] [`,` `current(`*varlist*`)` `lags(`$\#_0, [\#_1[, \ldots]]$`)`
`nosample` `static(`*varlist*`)` *other*-`regress`-*options*]

As with other estimation commands in Stata, typing `tsfit` by itself displays the last equation estimated by `tsfit`.

Four options distinguish `tsfit` from other regression routines in Stata: `current()`, `lags()`, `nosample`, and `static()`. The `nosample` option suppresses the display of information about the sample coverage. The other three options handle all the details needed to specify a time series regression.

`current(`*varlist*`)` identifies variables in the regression *varlist* for which a contemporaneous term is to appear as an explanatory variable, that is, variables whose $\beta_{i0} \neq 0$. The default is to include only lagged values of the explanatory variables.

`lags(`$\#_0, [\#_1[, \ldots]]$`)` specifies the number of lags of each of the variables to include in the regression. $\#_0$ specifies the length of $A(L)$, $\#_1$ specifies the length of $B_1(L)$, and so on. If fewer numbers than variables are specified, the last number specified is used for the remaining variables. If more numbers than variables are specified, the excess numbers are ignored.

`static(`*varlist*`)` specifies non-time series variables to include in the regression. These variables can also be incorporated by setting their maximum lag to zero using the `lags()` option. The `static()` option provides a more efficient means of incorporating these variables.

It may seem odd that `tsfit` excludes contemporaneous terms of the time series explanatory variables by default. Frequently, however, a time series regression is properly interpreted as a single equation from a vector autoregression or VAR, that is, a model of the form

$$\mathbf{A}(L)\mathbf{y}_t = \epsilon_t$$

where $\mathbf{y}_t$ is a vector and $\mathbf{A}(L)$ is a matrix polynomial in the lag operator. It is efficient to estimate a VAR one equation at a time if the lag lengths of all the variables are identical across equations. A single equation from a VAR has no current-dated explanatory variables; only lagged values appear. This equation can be handled by setting all the zero-th order coefficients of the lag polynomials to zero, that is, by setting

$$\beta_{i0} = 0, \quad i = 1, \ldots, J.$$

This specification arises so frequently in practice that `tsfit` makes it the default.

Example 1: Using tsfit to estimate the money-output correlation

You may find it easier to understand how `tsfit` works from an example than from a syntax diagram. One of the most thoroughly researched topics in economics is the so-called money-output correlation. An important unresolved question in this area is whether changes in the growth of the money stock can significantly affect the growth rate of real output. (Becketti and Morris 1992 discuss recent research in this area.) We will explore this question as a way of demonstrating the Stata programs discussed in this article.

The file `money.dta` contains quarterly data on the growth of real output as measured by gross domestic product (GDP), the growth of the money stock as measured by M2, inflation as measured by the growth in the GDP price deflator, short-term interest rates as measured by the secondary market yield on 3-month Treasury bills, and the spread between the rates on 6-month commercial paper and 6-month Treasury bills.

```
. use money, clear
(Growth rates for regression)

. describe

Contains data from money.dta
  Obs:   133 (max= 32766)                 Growth rates for regression
 Vars:     8 (max=    99)
Width:    28 (max=   200)
  1. year         int    %8.0g            Year
  2. quarter      int    %8.0g    quarter  Quarter
  3. date         float  %9.0g            Date
  4. D.rtb3       float  %9.0g            Change in 3-month T-bill rate
  5. G.defl       float  %9.0g            Inflation
  6. G.gdp        float  %9.0g            Growth of real GDP
  7. G.m2         float  %9.0g            Growth of M2
  8. rqual        float  %9.0g            6-month CP minus T-bill rate
Sorted by:  year  quarter
```

Note the variable names in this data set. They take the form `X.xxx`. The character before the period is an operator. The suffix after the period indicates the variable being operated on. `G` denotes the growth rate operator, `D` denotes the difference operator, and `L` denotes the lag operator. For example, `G.gdp` is the growth rate of real GDP. If digits appear between the operator and the period, they denote the power of the operator. For instance, `L3.x` is the third lag of the variable `x`. In *sts2*, I presented utility programs for calculating growth rates (`growth`), differences (`dif`), and lags (`lag`). These programs as well as other utility programs from *sts2* are included on the current STB diskette.

Let's begin by regressing output growth on money growth.

```
. regress G.gdp G.m2

  Source |       SS       df       MS                  Number of obs =     133
---------+------------------------------               F(  1,   131) =    6.96
   Model |  92.5815552     1  92.5815552               Prob > F      =  0.0093
Residual |  1742.27743   131  13.2998277               R-square      =  0.0505
---------+------------------------------               Adj R-square  =  0.0432
   Total |  1834.85898   132  13.9004468               Root MSE      =  3.6469

------------------------------------------------------------------------------
   G.gdp |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    G.m2 |   .2520064   .0955152      2.638   0.009       .0630546    .4409581
   _cons |   .9504947    .781776      1.216   0.226      -.5960448    2.497034
------------------------------------------------------------------------------

. period 4
4 (quarterly)

. datevars year quarter

. tsfit G.gdp G.m2, c(G.m2)
Quarterly data:  1959:2 to 1992:2    (133 obs)

  Source |       SS       df       MS                  Number of obs =     133
---------+------------------------------               F(  1,   131) =    6.96
   Model |  92.5815552     1  92.5815552               Prob > F      =  0.0093
Residual |  1742.27743   131  13.2998277               R-square      =  0.0505
---------+------------------------------               Adj R-square  =  0.0432
   Total |  1834.85898   132  13.9004468               Root MSE      =  3.6469

------------------------------------------------------------------------------
   G.gdp |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    G.m2 |   .2520064   .0955152      2.638   0.009       .0630546    .4409581
   _cons |   .9504947    .781776      1.216   0.226      -.5960448    2.497034
------------------------------------------------------------------------------
```

`regress` and `tsfit` produce identical results for this simple regression. Not surprisingly, `tsfit` uses `regress` to calculate the estimates. Since there are no lags in this equation (`lags(0)` is the default), we could have obtained the same results by typing

```
. tsfit G.gdp, s(G.m2)
```

There are a couple of other time series commands in this example that require explanation. `period` defines the period (frequency) of the data. In this example, we are using quarterly data. `period` was first introduced in *sts2*. It has been modified, however, and you should install the new version included on the STB-15 diskette. `datevars` specifies the variables that contain the date information. These commands allow `tsfit` to produce the information on sample coverage that appears as the first line of output after the `tsfit` command. As a convenience, the `findsmpl` command produces sample coverage information apart from any estimation.

```
. findsmpl G.gdp G.m2
Quarterly data:  1959:2 to 1992:2    (133 obs)
```

Use Stata's `help` command to get more information on `period`, `datevars`, and `findsmpl`.

This example regression shows an apparently strong relationship between money growth and output growth. According to this regression, a 1 percentage point increase in money growth is associated with a 0.25 percentage point increase in output growth in the same quarter. This estimate is significant at the 1 percent level: the p-value of the coefficient on money growth is 0.009.

Regressions of this form are generally useless in time series research. Money growth, particularly the growth of a broad aggregate like M2, is unlikely to be an exogenous variable in this context. Thus ϵ_t, the error term in this regression, is almost certainly correlated with money growth in period t. As a consequence, the estimate of the coefficient on money growth is biased and inconsistent.

One simple cure for this problem is to regress output growth on lagged money growth rather than on its contemporaneous value. Lagged money growth, while stochastic, is *predetermined*: money growth in period $t-1$ is unaffected by the regression disturbance in period t. We can estimate this regression using `lag` and `regress`:

```
. lag G.m2

. regress G.gdp LG.m2
  Source |       SS       df       MS                  Number of obs =     132
---------+------------------------------               F(  1,   130) =   18.19
   Model |  223.272273     1  223.272273               Prob > F      =  0.0000
Residual |   1595.8602   130  12.2758477               R-square      =  0.1227
---------+------------------------------               Adj R-square  =  0.1160
   Total |  1819.13247   131  13.8865074               Root MSE      =  3.5037

------------------------------------------------------------------------------
   G.gdp |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   LG.m2 |    .398955   .0935475      4.265   0.000       .2138824    .5840275
   _cons |  -.2017359   .7685655     -0.262   0.793      -1.722251    1.318779
------------------------------------------------------------------------------
```

Note how operators can be composed: `LG.m2` is the lag of the growth rate of M2. `tsfit` can estimate this equation in one step:

```
. tsfit G.gdp G.m2, l(0,1)
Quarterly data:  1959:3 to 1992:2    (132 obs)
  Source |       SS       df       MS                  Number of obs =     132
---------+------------------------------               F(  1,   130) =   18.19
   Model |  223.272273     1  223.272273               Prob > F      =  0.0000
Residual |   1595.8602   130  12.2758477               R-square      =  0.1227
---------+------------------------------               Adj R-square  =  0.1160
   Total |  1819.13247   131  13.8865074               Root MSE      =  3.5037

------------------------------------------------------------------------------
   G.gdp |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
   LG.m2 |    .398955   .0935475      4.265   0.000       .2138824    .5840275
   _cons |  -.2017359   .7685655     -0.262   0.793      -1.722251    1.318779
------------------------------------------------------------------------------
```

For this small model, there is little advantage in using `tsfit`. Models this small, however, are rare in time series analysis. Generally, many lags of the variables are included. The number of lags is frequently a multiple of the periodicity of the data. Four, eight, or twelve lags might be used with quarterly data, for example. `tsfit` makes this easy.

```
. tsfit G.gdp G.m2, lags(4)
Quarterly data:  1960:2 to 1992:2    (129 obs)
  Source |       SS       df       MS                  Number of obs =     129
---------+------------------------------               F(  8,   120) =    4.28
   Model |  395.856758     8  49.4820948               Prob > F      =  0.0001
Residual |  1386.61222   120  11.5551018               R-square      =  0.2221
---------+------------------------------               Adj R-square  =  0.1702
   Total |  1782.46898   128  13.9255389               Root MSE      =  3.3993

------------------------------------------------------------------------------
   G.gdp |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  LG.gdp |   .1922915   .0895322      2.148   0.034       .0150239    .3695591
 L2G.gdp |   .1853178   .0928391      1.996   0.048       .0015027    .3691328
```

```
     L3G.gdp |  -.0371747   .0916445     -0.406   0.686      -.2186244     .144275
     L4G.gdp |   .0321815   .0860394      0.374   0.709      -.1381705    .2025335
       LG.m2 |    .379388   .1263462      3.003   0.003       .1292313    .6295447
      L2G.m2 |   .0295384   .1527102      0.193   0.847      -.2728172    .3318939
      L3G.m2 |  -.0449752   .1523454     -0.295   0.768      -.3466085     .256658
      L4G.m2 |  -.0295819   .1324129     -0.223   0.824      -.2917502    .2325864
       _cons |  -.7991246   .9556427     -0.836   0.405      -2.691231    1.092981
------------------------------------------------------------------------------
```

When only one value is specified in the `lags()` option, that value is used for all the variables. Note also how the usable sample shrinks as more lags are added to the regression. The kth lag of any variable is a missing value for the first k observations. To incorporate four lags then, we lose the first four observations. The information displayed by `tsfit` makes clear the precise sample coverage of each equation.

tsmult: Calculating multipliers, permanent effects, and Granger tests

We are rarely interested in the individual coefficients in a time series regression. Instead, we pose questions about functions of the coefficients. Three questions we typically wish to answer are (1) does a temporary change in an explanatory variable have a statistically significant effect on the dependent variable, (2) does a permanent change in an explanatory variable have a statistically significant effect on the dependent variable, and (3) what is the predicted effect of a permanent change in an explanatory variable?

The first question is equivalent to asking whether all the coefficients on an explanatory variable are zero. Thus, if the variable of interest is x_{1t}, this question can be answered by testing the null hypothesis

$$\beta_{10} = 0, \beta_{11} = 0, \ldots, \beta_{1q} = 0$$

against the alternative that at least one of the coefficients is non-zero. When only lagged values of x_{1t} appear in the equation (when $\beta_{10} \equiv 0$), this test is called the Granger test for causality. The Granger test attempts to determine whether lagged values of x_{1t} improve predictions (reduce forecast errors) of y_t after controlling for lagged values of y_t and other explanatory variables. If so (if the null is rejected), x_{1t} is said to Granger-cause y_t.

The second question is equivalent to asking whether the *sum* of the coefficients on an explanatory variable is zero. Using x_{1t} for illustration again, this question can be answered by testing the null hypothesis

$$B_1(1) = \sum_{j=0}^{q} \beta_{1j} = 0$$

against the alternative

$$B_1(1) = \sum_{j=0}^{q} \beta_{1j} \neq 0.$$

Note that we can conveniently indicate the sum of coefficients by replacing L with 1 as the argument of the lag polynomial.

Answering the third question takes a bit more calculation. Consider a simple model with one explanatory variable:

$$A(L)y_t = \mu + B(L)x_t + \epsilon_t.$$

Now consider a permanent increase in x_t, that is, consider increasing x_t by, say, d units every period. Call the resulting series $\tilde{x}_t$. Then

$$\tilde{x}_t \equiv x_t + d.$$

To isolate the effects of this change, rewrite the model as

$$y_t = A^{-1}(L)\mu + A^{-1}(L)B(L)x_t + A^{-1}(L)\epsilon_t$$

where $A^{-1}(L)$ is the inverse of $A(L)$. Let $\tilde{y}_t$ be the series that results when x_t is replaced by $\tilde{x}_t$. By construction,

$$\tilde{y}_t = A^{-1}(L)\mu + A^{-1}(L)B(L)\tilde{x}_t + A^{-1}(L)\epsilon_t,$$

thus the change in y_t is

$$\tilde{y}_t - y_t = A^{-1}(L)B(L)(\tilde{x}_t - x_t) = A^{-1}(L)B(L)d.$$

The permanent effect of this d-unit change in x_t is

$$\begin{aligned}\lim_{t\to\infty}(\tilde{y}_t - y_t) &= A^{-1}(1)B(1)d \\ &= (B(1)/A(1))d \\ &= \frac{\sum_{i=0}^{q}\beta_i}{\sum_{j=0}^{p}\alpha_j}d.\end{aligned}$$

This permanent effect, the ratio multiplying d, is also called the long-run multiplier of x_t. This analysis assumes that x_t is purely exogenous. If x_t is endogenous—if, for instance, this is one equation from a VAR—there is no way to estimate the long-run multiplier of x_t from a single-equation model.

`tsmult` is designed to answer these three questions. The `tsmult` command takes no arguments or options. We demonstrate `tsmult` on another version of the money-output equation.

```
. tsfit G.gdp G.m2, l(1)
Quarterly data:  1959:3 to 1992:2    (132 obs)
  Source |       SS       df       MS                  Number of obs =     132
---------+------------------------------               F(  2,   129) =   13.43
   Model |  313.422927     2  156.711463               Prob > F      =  0.0000
Residual |  1505.70954   129   11.672167               R-square      =  0.1723
---------+------------------------------               Adj R-square  =  0.1595
   Total |  1819.13247   131  13.8865074               Root MSE      =  3.4165

------------------------------------------------------------------------------
   G.gdp |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  LG.gdp |   .2274921   .0818573      2.779   0.006       .0655354    .3894489
   LG.m2 |   .3409522   .0935755      3.644   0.000       .1558107    .5260936
   _cons |  -.4119275   .7532364     -0.547   0.585      -1.902224    1.078369
------------------------------------------------------------------------------

. tsmult
         |  Joint      Sum of              Long run
         | p-value  coefficients  p-value   effect
---------+------------------------------------------
G.gdp    |   0.01       0.773      0.00      1.294
G.m2     |   0.00       0.341      0.00      0.441
Last lag |   0.00
 Xs only |   0.00
---------------------------------------------------
```

The first column displays the p-value of the test that all the coefficients on a particular variable are zero; this is the answer to the first question above. The first row of the table displays the results for the lagged values of the dependent variable, the second row displays the results for the first time series explanatory variable, and so on. Thus, the p-value (rounded to two digits) for the test that all the coefficients on the lagged values of `G.gdp` are zero is 0.01. In this simple example, there is only one lagged value of `G.gdp` in the model, so you can easily compare the table produced by `tsmult` to the results of the regression.

The second column displays the sum of the coefficients on a particular variable. Recall that the lag polynomial for the dependent variable is written differently from the other lag polynomials, that is,

$$A(L) = 1 - \alpha_1 L - \alpha_2 L^2 - \cdots - \alpha_p L^p.$$

As a consequence, the sum of the coefficients for the dependent variable is

$$A(1) = \sum_{i=0}^{p}\alpha_i = 1 - \alpha_1 - \alpha_2 - \cdots - \alpha_p.$$

In this example, the coefficient on `LG.gdp` is 0.227, so the sum of coefficients is $1 - 0.227 = 0.773$. For other time series variables, the sum of coefficients is

$$B_i(1) = \sum_{j=0}^{q}\beta_{ij} = \beta_{i0} + \beta_{i1} + \cdots + \beta_{iq}.$$

In this example, there is only one term in the lag polynomial on money growth, so the sum of coefficients is just the coefficient on this variable, 0.341.

The third column displays the p-value for the test that the sums of the coefficients are zero; this is the answer to the second question above. In this example, the sums are highly significant for both output growth and money growth.

The fourth column displays the long-run effect on the dependent variable of a permanent one-unit increase in each of the variable; this is the answer to the third question above. For the dependent variable, the fourth column displays the long-run effect of a permanent one-unit increase in the regression disturbance, that is, the column displays

$$A^{-1}(1) = \frac{1}{A(1)} = \frac{1}{\sum_{i=0}^{p} \alpha_i}.$$

In this example, this effect is $1/.773 = 1.294$. For money growth, the long-run effect is $.341/.773 = .441$.

The last two rows of the table report the results of two other hypothesis tests of interest. The row labeled `Last lag` reports the p-value of the test that the coefficients on the last lags of all the variables including the dependent variable are all zero. If this null is accepted, all of the lag polynomials can be shortened by one term. The row labeled `Xs only` is the p-value for the same test but excluding the last lag of the dependent variable. Frequently we fix the length of the lag polynomial on the dependent variable to account for serial correlation in the left-hand-side variable. In this case, we are still interested in reducing the other lag polynomials to the shortest acceptable length.

tsreg: Putting it all together

So far, we have presented `tsfit` to calculate a time series regression and `tsmult` to calculate sums of coefficients, long-run multipliers, and to test some common hypotheses about the lag polynomials. These two commands occur together so frequently in practice that it is convenient to combine them in a single command.

`tsreg` combines `tsfit` and `tsmult` along with a new feature. As usual, a demonstration is clearer than an explanation.

```
. tsreg G.gdp G.m2, l(1)
Quarterly data:  1959:3 to 1992:2    (132 obs)

  Source |       SS       df       MS                  Number of obs =     132
---------+------------------------------               F(  2,   129) =   13.43
   Model |  313.422927     2  156.711463               Prob > F      =  0.0000
Residual |  1505.70954   129   11.672167               R-square      =  0.1723
---------+------------------------------               Adj R-square  =  0.1595
   Total |  1819.13247   131  13.8865074               Root MSE      =  3.4165

------------------------------------------------------------------------------
   G.gdp |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  LG.gdp |   .2274921   .0818573      2.779   0.006       .0655354    .3894489
   LG.m2 |   .3409522   .0935755      3.644   0.000       .1558107    .5260936
   _cons |  -.4119275   .7532364     -0.547   0.585      -1.902224    1.078369
------------------------------------------------------------------------------

         |  Joint      Sum of              Long run
         | p-value  coefficients  p-value   effect
---------+-----------------------------------------
G.gdp    |   0.01       0.773      0.00      1.294
G.m2     |   0.00       0.341      0.00      0.441
Last lag |   0.00
 Xs only |   0.00
---------------------------------------------------

              AIC: 2.48
             AICC: 3.497
              FPE: 11.937
               HQ: 2.506
              MDL: 7.465
               PC: 11.937
adjusted R-squared: .16
         R-squared: .17
 Schwarz criterion: 2.545
Durbin-Watson test: 2.06
  seasonal DW test: 1.804
        Q(4) test: .37
       LM(4) test: .32
     ARCH(4) test:  0.85
           H test: .84
   normality test: .01
```

`tsreg` calls `tsfit` to display information about the sample coverage and to estimate the regression. Next it calls `tsmult` to display the table of information about the lag polynomials. Finally it calls a new program, `regdiag`, to display a host of model selection criteria and diagnostic tests of the regression residuals.

`regdiag` can be used after any regression. Most of the criteria and test statistics it reports are equally applicable to cross section and time series regressions. For instance,

```
. quietly regress G.gdp G.m2
. regdiag, aic h norm
                  AIC: 2.603
               H test: .36
       normality test: .02
```

In this example, `regdiag` displays Akaike's information criterion (`AIC`), a model selection criterion, the *p*-value of Harvey's test (`H`) for heteroscedasticity, and the *p*-value of the Shapiro–Francia test of the normality of the regression residuals. `tsreg` calls `regdiag` with the `all` option, that is, it displays every statistic `regdiag` will calculate.

Because of space limitations, a full explanation of `regdiag` is deferred until the next issue of the STB. In the interim, you can type `help regdiag` for more information.

The syntax of `tsreg` is

`tsreg` [*varlist*] [`if` *exp*] [`in` *range*] [*weight*] [, `current(`*varlist*`)` `lags(`$\#_0$, [$\#_1$[, . . .]]`)` `nomult`
`noregress` `replace` `nosample` `static(`*varlist*`)` `notest` *other*-`regress`-*options*]

As usual, typing `tsreg` by itself displays the last equation estimated by `tsreg`.

Five options distinguish `tsreg` from `tsfit`. Four of the options—`nomult`, `noregress`, `nosample`, and `notest`—allow you to suppress the display of the `tsmult` table, the regression output, the sample coverage information, and the test statistics, respectively. The `noregress` option is particularly useful. When time series regressions contain long lag polynomials, the individual regression coefficients are generally uninterpretable. In these cases, attention is focused instead on the `tsmult` table and the diagnostic statistics.

An important difference between `tsreg` and `tsfit` is the treatment of the lagged variables included in the regression. `tsfit` leaves these newly created variables in the current data set. These variables must be present when other time series commands, such as `tsmult`, are used. Because `tsreg` combines the most commonly used commands, it restores the user's original data set when it is done. The `replace` option tells `tsreg` to leave the lagged variables in the data set so they can be used in later calculations.

An assessment and a look ahead

There are now enough time series and time-series-related commands in Stata that is difficult to remember them all. Table 1 lists most of them.

Stata's time series commands are highly interdependent. The `lag` command, for example, is fundamental. It is called by many of the other commands, and the `lag` command's convention for denoting operators is assumed and exploited by other routines. Moreover, time series commands tend to save more results in system macros for the use of later commands. The saved results of `tsfit` and `tsmult` are so extensive, for instance, that I abandoned any attempt to document them in this article.

Despite the considerable progress documented in the table above, much remains to be done to complete Stata's battery of time series commands. Most pressing is the need for a command to calculate dynamic forecasts and simulations of a time series regression. In fact, I have written such a command, `tspred`. Because of space limitations and because this article is already so long, I am deferring the presentation of `tspred` to the next issue of the STB. Other important extensions of the single-equation approach are rolling regressions and CUSUM tests of parameter stability. Again, I have written commands that provide these techniques, and they will appear in future issues of the STB.

Other important commands have not yet been written. Most important of these are commands to estimate and manipulate vector autoregressive models, VARs. The Johansen approach to estimating and testing cointegrated models needs to be incorporated in the VAR commands. Commands to estimate Box–Jenkins time series models, ARCH and GARCH models, the Kalman filter, and dynamic factor models are all needed and are on the development schedule.

Because these time series commands are under active development, your comments and suggestions are particularly helpful. Use these commands, then let me know which of their features you like and which you would change.

Table 1

Command	Documentation	Description
`ac`	*sts1*	autocorrelation plot
`coint`	*sts2*	Engle-Yu cointegration test
`corc`	[5s] corc	regression with serial correlation correction
`datevars`	*sts4*	specify date variables
`dickey`	*sts2*	unit root tests
`dif`	*sts2*	generate differences
`dropoper`	*sts2*	drop operator variables
`findlag`	*sts2*	find optimal lag length
`findsmpl`	*sts4*	display sample coverage
`growth`	*sts2*	generate growth rates
`ksm`	[5s] ksm	smoothing including lowess
`lag`	*sts2*	generate lags
`lead`	*sts2*	generate leads
`pac`	*sts1*	partial autocorrelation plot
`period`	*sts2*	specify period (frequency) of data
`regdiag`	*sg20*	display regression diagnostics
`runtest`	[5s] runtest	test for random order
`smooth`	[5s] smooth	robust nonlinear smoother
`tsfit`	*sts4*	estimate time series regression
`tsmult`	*sts4*	information about lag polynomials
`tsreg`	*sts4*	combined `tsfit`, `tsmult`, and `regdiag`
`xcorr`	*sts3*	cross correlations

References

Becketti, S. and C. Morris. 1992. Does money matter anymore?—A Comment of Friedman and Kuttner. Research Working Paper RWP 92–07. Federal Reserve Bank of Kansas City.