STATA TECHNICAL BULLETIN

May 1995
STB-25

A publication to promote communication among Stata users

Subscriptions are available from Stata Corporation, email stata@stata.com, telephone 979-696-4600 or 800-STATAPC, fax 979-696-4601. Current subscription prices are posted at www.stata.com/bookstore/stb.html.

Previous Issues are available individually from StataCorp. See www.stata.com/bookstore/stbj.html for details.

Submissions to the STB, including submissions to the supporting files (programs, datasets, and help files), are on a nonexclusive, free-use basis. In particular, the author grants to StataCorp the nonexclusive right to copyright and distribute the material in accordance with the Copyright Statement below. The author also grants to StataCorp the right to freely use the ideas, including communication of the ideas to other parties, even if the material is never published in the STB. Submissions should be addressed to the Editor. Submission guidelines can be obtained from either the editor or StataCorp.

Contents of this issue — page

an54.	STB-19–STB-24 available in bound format	2
dm28.	Calculate nice numbers for labeling or drawing grid lines	2
dm29.	Create TEX tables from data	3
dm30.	Comparing observations within a data file	7
sg26.3.	Fractional polynomial utilities	9
sg32.1.	Variance inflation factors and variance-decomposition proportions: Correction	13
sg35.	Robust tests for the equality of variances	13
sg36.	Tabulating the counts of multiple categorical variables	15
sg37.	Orthogonal polynomials	17
sg38.	Generating quantiles	19
sg39.	Independent percentages in tables	20
snp8.	Robust scatterplot smoothing: enhancements to Stata's `ksm`	23
sts10.	Prais–Winsten regression	26
zz5.	Cumulative index for STB-19–STB-24	29

an54 STB-19—STB-24 available in bound format

Sean Becketti, Stata Technical Bulletin, stb@stata.com, FAX 914-533-2902

The fourth year of the *Stata Technical Bulletin* (issues 19–24) has been reprinted in a 242+ page bound book called *The Stata Technical Bulletin Reprints, Volume 4*. The volume of reprints is available from StataCorp for $25–$20 for STB subscribers—plus shipping. Authors of inserts in STB-19–STB-24 will automatically receive the book at no charge and need not order.

This book of reprints includes everything that appeared in issues 19–24 of the STB. As a consequence, you do not need to purchase the reprints if you saved your STBs. However, many subscribers find the reprints useful since they are bound in a volume that matches the Stata manuals in size and appearance. Our primary reason for reprinting the STB, though, is to make it easier and cheaper for new users to obtain back issues. For those not purchasing the reprints, note that *zz5* in this issue provides a cumulative index for the fourth year of the original STBs.

dm28 Calculate nice numbers for labeling or drawing grid lines

James W. Hardin, Stata Corporation, FAX 409-696-4601, EMAIL stata@stata.com

nicenum computes lists of “nice” numbers (multiples of 2, 5, and 10) that can be used as the arguments to the **xlab**, **ylab**, **xline**, and **yline** options of the **graph** command. The syntax of **nicenum** is

nicenum *macroname* = *arglist* [, **n**umber(#)]

The *arglist* can contain variables, scalars, and numeric constants. **nicenum** computes a list of nice numbers that cover the range of values specified by the *arglist* and stores that list, separated by commas, in *macroname*. The **number()** option specifies a desired number of values in the list of nice numbers. **nicenum** regards that number as a suggestion rather than a constraint; the program will alter the number as needed to produce a list of numbers that is acceptably nice.

Remarks

The axis-labeling options to the **graph** command automatically choose nice numbers if you do not specify values. For instance, if you type the commands

```
. use auto
. graph mpg price, xlabel ylabel
```

where **auto** is the automobile data supplied with Stata, the x-axis of the graph will be labeled with the values 0, 5,000, 10,000, and 15,000 and the y-axis will be labeled with the values 10, 20, 30, and 40.

nicenum calculate similar lists of numbers. In addition, by adding values to the *arglist*, you can force the list of numbers to include specified values. This feature is particularly useful in do-files and Stata programs, when you want to ensure that a sequence of graphs uses the same scale and labels.

Example

```
. use auto
(1978 Automobile Data)

. summarize price

Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
   price |      74    6165.257   2949.496       3291      15906

. nicenum prlab = price

. display "$prlab"
0,5000,10000,15000,20000

. by foreign: summarize price

-> foreign=Domestic
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
   price |      52    6072.423   3097.104       3291      15906

-> foreign= Foreign
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
   price |      22    6384.682   2621.915       3748      12990
```

```
. nicenum forlab = price if foreign
. display "$forlab"
2000,4000,6000,8000,10000,12000,14000
. nicenum forlab = price if foreign, number(3)
. display "$forlab"
0,5000,10000,15000
. nicenum forlab = price 20000 if foreign
. display "$forlab"
0,5000,10000,15000,20000
```

dm29 Create TEX tables from data

James W. Hardin, Stata Corporation, FAX 409-696-4601, EMAIL stata@stata.com

The syntax of **textab** is

textab *varlist* [**if** *exp*] [**in** *range*] [, **vlin**es(*string*) **sep**(*string*) **tab**skip(*string*) **f**ont(*string*) **bstr**s(*string*) **estr**s(*string*) **miss**ing(*string*) **al**ign(*string*) **form**at(*string*) **nocen**ter]

textab generates TEX code to format a table containing the values of the variables in the *varlist*. This code can be used in a plain TEX or LATEX document.

Options

vlines(*string*) is used to specify the types and positions of vertical rules in the table. In a table with k columns, there are $k+1$ possible positions for vertical rules: on the left-hand side of the table, between the first and second columns, between the second and third columns, ..., between the $k-1$ and k columns, and on the right-hand side of the table. The argument of the option is a comma-separated string with $k+1$ entries. Each entry is a single letter where **s** denotes a single line, **b** denotes a bold line, **d** denotes a double line, and **n** denotes no line. The default is that all $k+1$ entries are equal to **n**.

sep(*string*) is used to specify the types of horizontal lines to be drawn in the table. The argument of the option is a comma-separated string with three entries that specify the line at the top of the table, the line beneath the column headers, and the line below the table, respectively. Each entry is a single letter where **s** denotes a single line, **b** denotes a bold line, and **n** denotes no line. The default is to not draw any horizontal lines.

tabskip(*string*) is used to specify the space between each column of information. There are $k+1$ tabskips in a table: the amount of space to skip before starting the table, the amount of space to place between each column of information, and the amount of space to the right of the last column. The default is to skip **0pt** to the left of the table and then to have **10pt** of space between each column. The argument to this option is a comma-separated list of numbers that indicate the skips in terms of printer's points.

font(*string*) is used to specify fonts for each column of information. This is a comma-separated list of arguments that will be placed directly into the **halign** template so it is up to you to specify the entire font (e.g., **\rm**). The default is to not specify any fonts.

bstrs(*string*) is used to specify a string that should appear in every entry in a column at the beginning of the entry. This is a comma-separated list and will be placed in curly braces in the **halign** template outside of any font that you may have specified. You may specify a font directly in this argument if you wish. There are k entries in this list. Should you want to include a space before or after one of these entries, then enclose the entire string in curly braces.

estrs(*string*) is used to specify a string that should appear in every entry in a column at the end of the entry. This is a comma-separated list and will be placed in curly braces in the **halign** template outside of any font that you may have specified. You may specify a font directly in this argument if you wish. There are k entries in this list. Should you want to include a space before or after one of these entries, then enclose the entire string in curly braces.

`missing(`*string*`)` is a comma-separated list specifying a string to be placed into the table for any missing value that is encountered when creating the table. There are k entries in this list. The default is not to make any substitution.

`align(`*string*`)` is used to specify the horizontal alignment of each of the columns. This is a comma-separated list of k entries. Each entry is a single letter where `l` denotes a flush-left alignment, `c` denotes a centered alignment, and `r` denotes a flush-right alignment. The default is to center all columns.

`format(`*string*`)` is used to format strings or numbers before they are placed into the table. This is a comma-separated list of Stata format codes (e.g., %6.3f) for each of the k entries. The default is to use the values as they would be formatted by `list`.

`nocenter` specifies that the generated table should not contain TEX code to center the table horizontally. The default is to create a table that is centered.

Note: Stata does not handle the backslash character when the command line is parsed. If you need to pass an argument that includes one or more characters that require the backslash (such as positioning a '`$`' character), then you will have to directly edit the resulting TEX code yourself as there is no way for the command to access that character in any kind of passed string.

Description

If you use the TEX typesetting program to create documents, then you understand the difficulty of creating tables. The code and examples that come with the TEXbook are not very enlightening and most people agree that creating tables is a very difficult task. However, this need not be so. There are a few general and easy-to-remember rules to follow when creating a TEX table. The `textab` program knows these rules and creates tables that can be imported directly into a document with little or no further editing. The only editing that may be necessary is to escape characters that are special to TEX (see the previous note) or to add more headers or footers to the table. In the following section, I present the basic algorithm for designing a table in TEX and show how that algorithm is used in `textab` creates tables. All the tables in this article were created using the `textab` program.

This article will not attempt to explain every single aspect of creating tables in TEX. However, each table created by `textab` contains numerous comments. The output to the screen is also color coded to make seeing it much easier for those with color monitors. Each of the k columns and each row is clearly marked with a comment as are the struts and headers. Each strut is further commented to point out which of the 4 types of struts is being used. I go into further detail on struts in the following section. Finally, I include comments in the `halign` template section to clearly mark the separators and each of the k variables included in the table. Included in the comment for each of the variables is the alignment being used. The color coding is such that comments appear in green (row markers are in blue to more easily find them), and table values are in white. All TEX code appears on the screen in yellow.

Tables in TEX

Tables are created in TEX using the `halign` command. This command creates a template for the justification, font, size, and spacing of each column in the table. This template should define the following items:

Column	Purpose
Strut	Sets the height for the current row
Outside Vbar	Left-hand vertical rule
Value 1	First column of table values
Vbar	Vertical rule between table values
Value 2	Second column in table values
Vbar	Vertical rule between table values
...	...
Value k	Last column of table values
Outside Vbar	Right-hand vertical rule

The above table was produced by reading the string values into Stata from a file and issuing the command:

```
. textab Column Purpose, sep(s,s,s) vlines(s,s,s) align(l,l)
```

This shows that the `halign` template should define $2k+2$ items to typeset k columns of information. It should also be noted that this description of a table applies regardless of whether you actually want the vertical rules to be drawn. If you do not specify vertical rules, then these widths will be 0 and will take up no space in the final table. The first column is the strut and serves a special purpose in the table. The strut is a vertical rule of width 0, which makes it is invisible and prevents it from taking up any width in the table. Its purpose is to define the height of the current row. The `textab` command will adjust these heights depending on whether there are nearby horizontal rules in the table. It also defines the strut in terms of the baseline so that it will work regardless of the current font or magnification that you are using in your document. There are only 3 horizontal rules that `textab` will draw for you. One is above the entire table, the next is below the column headers, and the last is below the entire table. This leads to a need for 4 different struts. A normal strut is one in which there is no horizontal rule either above or below the current row. There may also be a row which has a horizontal rule above it, but not one below it. There may be a row with a horizontal rule below it, but not above it. Finally, there may be a row which has a horizontal rule both above and below it. Each of these types of struts appear in the preceding table. The column headers have a horizontal rule above and below. The first row of the table has a horizontal rule above. The last row of the table has a horizontal rule below, and all other rows do not have neighboring horizontal rules.

The final point of consideration is the amount of space between each column of information in the table. TeX provides the `tabskip` command to allow you to specify this space in the `halign` template. There are $k+1$ tabskips to be specified to the `textab` command which are then placed into the `halign` template. The first `tabskip` is the one that specifies how far from the left margin to skip before starting the table. If you are going to center the table, then you can specify this as zero (which is the default). The remaining `tabskip` values are split evenly on either side of the `vrule` columns. Should you want uneven `tabskip` spacing, you will have to edit the table that `textab` generates. You will also have to edit the resulting table if you would like a more descriptive column header than is possible from the eight characters allowed in the variable name.

Examples

In order to demonstrate all of the options of the `textab` command, I will use the same small data set for all of the following examples. This contrived data set has both numeric and string variables and will be used to make tables that have no value other than demonstrating various properties of the `textab` command.

```
. list
            Name        Test1       Rank       Value
1.          John       89.992          3           A
2.          Bill       71.023          5           C
3.          Mary            .          2           A
4.         Janet       80.923          4           B
5.       William       94.556          1           A
```

Here are some sample tables and the commands that created them.

Name	Test1
John	89.992
Bill	71.023
Mary	*absent*
Janet	80.923
William	94.556

Name	Test1	Value
John	89.992	A
Bill	71.023	C
Mary	*absent*	A
Janet	80.923	B
William	94.556	A

Name	Test1	**Value**
John	89.992	**A**
Bill	71.023	**C**
Mary	*absent*	**A**
Janet	80.923	**B**
William	94.556	**A**

```
. textab Name Test1, missing(,\it absent) sep(s,s,s) nocenter
. textab Name Test1 Value, missing(,\it absent) vlines(s,d,s,s) nocenter
. textab Name Test1 Value, miss(,\it absent) sep(s,b,s) vlin(s,d,s,s) nocen al(l,c,c) font(,,\bf)
```

Name	$\mathcal{R}$−Rank
John	$\mathcal{R}$−3
Bill	$\mathcal{R}$−5
Mary	$\mathcal{R}$−2
Janet	$\mathcal{R}$−4
William	$\mathcal{R}$−1

Name	Average	Rank	Value
John	89.99	3	A
Bill	71.02	5	C
Mary	92.28	2	A
Janet	80.92	4	B
William	94.56	1	A

```
. textab Name Rank, sep(s,b,s) vlines(s,d,s) bstr(,$\cal R-$) tabskip(0,20,40) nocenter
. generate Average = Test1
. replace Average = 92.28 in 3
. textab Name Average Rank Value,sep(s,s,s) vlines(s,s,s,s,s) format(,%5.2f,,) nocenter
```

Finally, here is an example that illustrates the need for the 4 struts illustrated earlier in the text. If I typeset the last table again but set all struts to the usual value, I obtain

Name	Average	Rank	Value
John	89.99	3	A
Bill	71.02	5	C
Mary	92.28	2	A
Janet	80.92	4	B
William	94.56	1	A

Below, I present the TEX code that was generated by the **textab** command for the "correct" version of this table. The "incorrect" table was created by changing the **strut(A)**, **strut (B)**, and **strut(AB)** lines to be the same as the **strut** line (which is the default behavior in TEX, but not in **textab**.)

```
% BEGINNING OF TEXTAB TABLE
\vbox{
  \tabskip=0pt%                                                  Tab0
\halign{
  #\tabskip=0pt&% strut with width=0pt for vertical bars if they exist
  #\tabskip=5pt&%                                                (Sep)
  {\hfil}{{#}}{\hfil}\tabskip=5pt&%                                (C) Var 1
  #\tabskip=5pt&%                                                (Sep)
  {\hfil}{{#}}{\hfil}\tabskip=5pt&%                                (C) Var 2
  #\tabskip=5pt&%                                                (Sep)
  {\hfil}{{#}}{\hfil}\tabskip=5pt&%                                (C) Var 3
  #\tabskip=5pt&%                                                (Sep)
  {\hfil}{{#}}{\hfil}\tabskip=5pt&%                                (C) Var 4
  #\tabskip=0pt\cr%                                              (Sep)
%
% End of halign directive and beginning of column headers
%
\noalign{\hrule}
    \vrule height 1.1\baselineskip depth 0.7\baselineskip width0pt&% strut (AB)
    {\vrule}&Name&%
    {\vrule}&Average&%
    {\vrule}&Rank&%
    {\vrule}&Value&%
    {\vrule}\cr%
%
% End of headers and beginning of table values
%
\noalign{\hrule}
    \vrule height 1.1\baselineskip depth 0.3\baselineskip width0pt&% strut (A)
    {\vrule}&John&%                                     Column 1
    {\vrule}&89.99&%                                    Column 2
    {\vrule}&3&%                                        Column 3
    {\vrule}&A&%                                        Column 4
    {\vrule}\cr%                                        Row 1
    \vrule height 0.7\baselineskip depth 0.3\baselineskip width0pt&% strut
    {\vrule}&Bill&%                                     Column 1
    {\vrule}&71.02&%                                    Column 2
    {\vrule}&5&%                                        Column 3
    {\vrule}&C&%                                        Column 4
    {\vrule}\cr%                                        Row 2
    \vrule height 0.7\baselineskip depth 0.3\baselineskip width0pt&% strut
    {\vrule}&Mary&%                                     Column 1
```

```
    {\vrule}&92.28&%                                      Column 2
    {\vrule}&2&%                                          Column 3
    {\vrule}&A&%                                          Column 4
    {\vrule}\cr%                                          Row 3
    \vrule height 0.7\baselineskip depth 0.3\baselineskip width0pt&% strut
    {\vrule}&Janet&%                                      Column 1
    {\vrule}&80.92&%                                      Column 2
    {\vrule}&4&%                                          Column 3
    {\vrule}&B&%                                          Column 4
    {\vrule}\cr%                                          Row 4
    \vrule height 0.7\baselineskip depth 0.7\baselineskip width0pt&% strut (B)
    {\vrule}&William&%                                    Column 1
    {\vrule}&94.56&%                                      Column 2
    {\vrule}&1&%                                          Column 3
    {\vrule}&A&%                                          Column 4
    {\vrule}\cr%                                          Row 5
\noalign{\hrule}
}}%               End of textab produced table
% END OF TEXTAB TABLE
```

References

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

von Bechtolsheim, S. 1993. *TEX in Practice Volume IV: Output Routines, Tables*. New York: Springer-Verlag.

dm30	Comparing observations within a data file

Richard Goldstein, Qualitas, Inc., EMAIL richgold@netcom.com

Stata includes commands for comparing a pair of variables within a file ([5d] compare) and for comparing a list of variables across two files ([5d] cf), but there is no easy way to compare two *observations* within a data set. Yet, especially when obtaining data from others, this is necessary to ensure against duplicate observations in the data set. I have written `compobs` to perform this task. The syntax of `compobs` is

`compobs` *varlist* `if _n==`*#* [, `list(`*varlist*`)` `number(`*#*`)`]

`compobs` expects that the data will be sorted in some way that eases the search for such duplicates (e.g., sorted by some external identifying number such as Social Security number). As a consequence, `compobs` examines adjacent observations by default. The observations to examine are specified by the `if` clause, which is required. Replace the '*#*' with the number of the second observation of the pair to be compared. For example, to compare the values of all variables across observations 7 and 8, type

```
. compobs _all if _n==8
```

The output is presented in two parts: (1) the values of the variables that differ across observations are displayed; and (2) the number of variables with differences is displayed. The `list` option allows you to attach identifiers to each difference. The `number()` option allows you to compare the observation selected with the `if` clause to any other observation, not just the preceding observation. Just specify the desired observation number in this option. The observation number must, of course, be an integer.

Example

To demonstrate `compobs`, I use the familiar automobile data. First, I compare the values of several of the variables across the first two observations:

```
. use auto
(1978 Automobile Data)
. compobs price mpg rep78 hdroom weight if _n==2
        price
  1.     4099
  2.     4749
```

```
         mpg
  1.      22
  2.      17

      hdroom
  1.     2.5
  2.     3.0

      weight
  1.    2930
  2.    3350

Number of Differences =  4
```

Note that **compobs** did not display the variable **rep78** because the values were identical across observations.

```
. list rep78 in 1/2
       rep78
  1.       3
  2.       3
```

Now I use the **list()** option to display the make of each car along with the differences.

```
. compobs price mpg rep78 hdroom weight if _n==2, list(make)
       price                make
  1.    4099         AMC Concord
  2.    4749           AMC Pacer

         mpg                make
  1.      22         AMC Concord
  2.      17           AMC Pacer

      hdroom                make
  1.     2.5         AMC Concord
  2.     3.0           AMC Pacer

      weight                make
  1.    2930         AMC Concord
  2.    3350           AMC Pacer

Number of Differences =  4
```

Finally, I compare the values of all the variables in the second and the sixth observations.

```
. compobs _all if _n==2, number(6)
                      make
  2.             AMC Pacer
  6.         Buick LeSabre

       price
  2.    4749
  6.    5788

         mpg
  2.      17
  6.      18

      hdroom
  2.     3.0
  6.     4.0

       trunk
  2.      11
  6.      21

      weight
  2.    3350
  6.    3670

      length
  2.     173
  6.     218

        turn
  2.      40
  6.      43

       displ
  2.     258
  6.     231

      gratio
  2.    2.53
  6.    2.73

Number of Differences =  10
```

sg26.3	Fractional polynomial utilities

Patrick Royston, Royal Postgraduate Medical School, London, FAX (011)-44-181-740-3119

In this insert, I describe three utilities designed to enhance the `fp` (fractional polynomial) software of Royston and Altman (1994a,b). They are called `fpshow`, `fpplot` and `fpderiv`. The STB-25 diskette includes them and the most recent version of the suite of FP programs. See *sg26* for an explanation of fractional polynomials and their uses.

fpshow

The syntax for `fpshow` is

`fpshow` [, model(#) best info monotonic devdiff(#)]

`fpshow` gives extra information on the regression models that `fp` has (unless `fp`'s `log` option has been used) silently fitted to your data. This includes a comparison of the deviance of the best-fitting model with that of each of the other candidates fitted and an indication, where possible, if each model function is monotonic in X, the argument of the fractional polynomial.

Options

`model(#)` displays the results from model number # and makes that model the current one. `fp` numbers the models it fits from 1 to N, where N depends on the degrees of freedom specified in `df()` and on the number of powers in `powers()` (see `help fp`). If `fpshow` is subsequently typed without options, results from model # will be displayed again. Also, `fpgraph`, `fpplot` and other FP utilities will 'see' this model as the current one, so you can investigate it further. However, typing `fp` without a list of variables will always display results and comparisons from the best-fitting model (which is not necessarily the current one).

`best` displays results from the best-fitting model and makes that model the current one.

`info` gives the following information about each model fitted: its number (from 1 to N); the powers used in the fractional polynomial; whether the curve is monotonic (strictly increasing or decreasing over the entire range of X); the deviance; and the increase in deviance over the best-fitting model. Monotonicity cannot easily be determined from the model formula for models with degree m greater than 2 (which can occur if the `fixpowers()` option is used with `fp`); it is shown as '`--`' in these cases.

`monotonic` displays only models known to be monotonic. Note that some models with degree>2 are monotonic; these won't be indicated as such. See the comment in the `info` option.

`devdiff(#)` displays results only for the worst-fitting models, that is, those whose deviance is at least # greater than that of the best-fitting model. If # is negative, `devdiff()` displays only the best-fitting models, those whose deviance is no more than minus # greater than that of the best-fitting model. A sensible value of # is 4 (or -4).

Example

As an example I shall use a data set, `igg.dta`, that contains data on IgG (immunoglobulin-G), a protein important in the human immune response. This file was originally supplied on the STB-21 diskette and is reproduced on the STB-25 diskette.

```
. use igg
. describe
Contains data from \a\c38\igg.dta
  Obs:   298 (max=  2278)
 Vars:     3 (max=    99)
Width:    12 (max=   200)
  1. igg          float  %9.0g                IgG (g/l)
  2. age          float  %9.0g                Age (years)
  3. y            float  %9.0g                Square root of IgG
Sorted by:
```

These data were recorded on 298 children between 6 months and 6 years old. Here I shall model the mean of y, the square root of IgG, as a function of age. The square-root transformation approximately normalizes the distribution of IgG and stabilizes its variance. For physiological reasons, IgG is expected to increase monotonically with age, so we will probably reject models which don't have this feature. First we fit FP models of degree $m = 2$:

```
. fp y age
MODELS, POWERS (p), DEVIANCES (D) and GAINS (G) for Y = y, X = age.
(*) Base model  Linear   Quadratic     Cubic     BoxTid      df(2)      df(4)
------------------------------------------------------------------------------
p      --          1       1, 2     1, 2, 3      1, 1      0         -2,   2
D   427.539    337.561    333.884    327.687    331.294    327.436    319.448
G               0.000      3.677      9.874      6.267     10.125     18.113
Curve (-2,2) has a positive slope and no maximum or minimum for X>0.
(*) Base model = [none] (298 obs.)
. fpshow, info
Model #  Powers  Monotonic? Deviance    Dev.diff.
-------------------------------------------------
  1      -2          Yes     346.990     27.542
  2      -2,-2        No     334.921     15.472
  3      -2,-1        No     330.324     10.875
  4      -2,-.5      Yes     327.648      8.199
  (output omitted)
```

A total of 44 models were fitted—only the first four are shown above. We shall look at the fit of the best model later. First we use **fpshow** to inspect the models that are closest in deviance to the best one, then those that are monotonic:

```
. fpshow, devdiff(-4)
Model #  Powers  Monotonic? Deviance    Dev.diff.
-------------------------------------------------
  6      -2,.5       Yes     322.747      3.298
  7      -2,1        Yes     321.025      1.577
  8      -2,2        Yes     319.448      0.000 * +
  9      -2,3        Yes     319.844      0.396
 16      -1,2        Yes     321.714      2.266
 17      -1,3        Yes     320.964      1.515
 24      -.5,3       Yes     323.341      3.892
Current model (+); model with lowest deviance (*).
. fpshow, mono
Model #  Powers  Monotonic? Deviance    Dev.diff.
-------------------------------------------------
  1      -2          Yes     346.990     27.542
  4      -2,-.5      Yes     327.648      8.199
  5      -2,0        Yes     325.025      5.577
  6      -2,.5       Yes     322.747      3.298
  7      -2,1        Yes     321.025      1.577
  8      -2,2        Yes     319.448      0.000 * +
  (output omitted)
```

In fact, most of the models with $m = 2$ are monotonic, including (for example) model 7, which has powers $(-2, 1)$ and which is linear in X for large X. We make model 7 the current one:

```
. fpshow, model(7)
Model number  7
---------------
  Source |       SS       df       MS                Number of obs =     298
---------+------------------------------             F(  2,   295) =   63.37
   Model |  22.0143273     2  11.0071636             Prob > F      =  0.0000
Residual |  51.2380196   295  .173688202             R-squared     =  0.3005
---------+------------------------------             Adj R-squared =  0.2958
   Total |  73.2523469   297  .246640898             Root MSE      =  .41676

------------------------------------------------------------------------------
       y |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     X_1 |  -.1274086   .0310547     -4.103   0.000      -.1885254  -.0662917
     X_2 |   .1000895   .0193134      5.182   0.000       .0620799    .138099
   _cons |   2.048706   .0721687     28.388   0.000       1.906676   2.190737
------------------------------------------------------------------------------
Deviance =   321.025. Fractional power(s) used: -2,1.
Curve (-2,1) has a positive slope and no maximum or minimum for X>0.
```

We could now use fpgraph or fpplot (see below) to plot the fit of model 7 and compare it with that of the best model (model 8)—the fits are about equally good. To return to the best model, we would type fpshow, best or fpshow, model(8).

fpplot

fpplot supplements fpgraph (already provided by Royston and Altman 1994), providing a smooth plot of the fitted FP function or of an arbitrary FP function over a specified range of X values. The syntax for fpplot is

fpplot [, from(#) to(#) obs(#) generate(*xvar yhat*) nograph scale *graph_options*]

or

fpplot [, from(#) to(#) obs(#) generate(*xvar yhat*) nograph scale powers(*powlist*) coeffs(*coefflist*) constant(#) expx(#) *graph_options*]

Options

from(#) and to(#) define the lower and upper limits of X, respectively. If an FP model has recently been estimated, from() and to() are taken by default as the minimum and maximum of the X-values; otherwise, each # must be supplied.

obs(#) is the number of equally spaced values of X to be used; # must be between 2 and 500. Default: 100.

powers() is the set of fractional powers for the FP function. In the first form of fpplot, the program will determine the powers from the current value of the macro $S_E_pwrs, so you need not specify them; in the second form, you must supply them in *powlist*.

coeffs() is the set of coefficients (multipliers). The fitted function is of the form

$$\widehat{Y} = \beta_0 + \beta_1 H_1(X) + \beta_2 H_2(X) \ldots$$

where the H's are functions of X defined by the fractional powers. In the first form of fpplot, the coefficients and constant are provided by Stata's _b[] functions, so you need not specify them. In the second form, you must supply the coefficients in *coefflist* and the constant (if required) in constant().

constant(#) is the constant term (see coeffs() above).

expx(#) transforms X to $\exp(-\# * X)$ before calculating the FP function. In the first form of fpplot, # is taken from the macro $S_E_xpx so you need not specify it. Note that the untransformed values of X are always used in the plot, even when the exponential transformation has been applied.

generate() adds two new variables to the data: *xvar*, containing the values of X, and *yhat*, containing the values of the calculated FP function. If obs() exceeds the original number of observations, the dataset is enlarged accordingly.

scale linearly transforms Y to the range $[\,0, 1\,]$. This can be useful if several plots are to be superimposed.

nograph suppresses the plot.

graph_options refers to any of the options of the graph, twoway command.

Example

Having fitted a FP with powers $(-2, 2)$ to the IgG data, we can use the command

```
fpplot, from(0.5) to(10)
```

to plot the fit between 6 months and 10 years (the original range was 0.5–6 years). The result is shown as Figure 1.

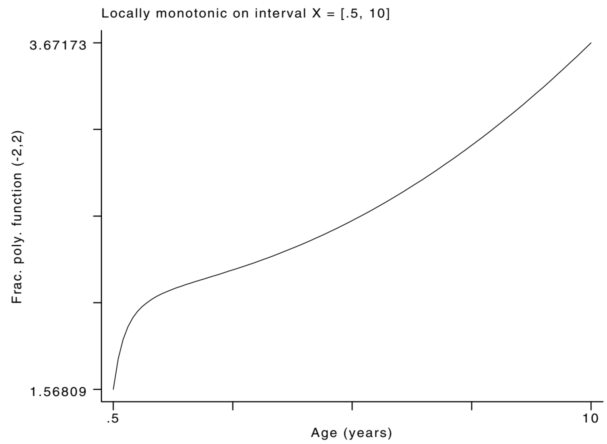

Figure 1: Fractional polynomial function fitted to IgG data

The coefficients of X^{-2}, X^2 and the constant for this fit were –0.1562, 0.0148 and 2.189, respectively. We could have used the alternative syntax and typed

```
fpplot, from(0.5) to(10) powers(-2 2) coeffs(-0.1562 0.0148) constant(2.189)
```

to achieve the same result. `fpplot` reports that the function is locally monotonic on the specified interval. In fact, we know from before that it is globally monotonic.

fpderiv

`fpderiv` calculates derivatives of FP functions, either the one most recently fitted or an arbitrary one. The syntax is

`fpderiv` *deriv_var* [, `powers(`*powlist*`)` `coeffs(`*coeff_list*`)` `next` `curvature` `dcurvature`]

`fpderiv` calculates the (analytic) first derivative of the FP function associated with the most recently fitted FP model and places the result into a new variable, *deriv_var*. All derivatives of FPs are in fact themselves FPs with powers differing from those of the original function. Higher derivatives may be obtained by repeated use of the `next` option. `fpderiv` also calculates a measure of the curvature of the function and the derivative of this measure (see `curvature` and `dcurvature` options).

Options

`powers()` defines the powers of the FP function. The default *powlist* is that used with the most recent FP model (and stored in `$S_E_pwrs`).

`coeffs()` defines the regression coefficients of the FP model. The default *coeff_list* is that estimated with the most recent FP model, and stored in Stata's `_b[]` functions. Note that your own *coeff_list* must be a 1 by m matrix, that is, a row vector of length m, where m is the degree of the FP function. For direct input, this simply amounts to a list of numbers separated by space(s). Note that *coeff_list* does *not* include the constant term `_b[_cons]`, as this plays no part in calculating the derivative.

`next` finds the next higher derivative. For example, if you just calculated the first derivative by using `fpderiv` without options, `fpderiv d2, next` would put the second derivative into `d2` and then `fpderiv d3, next` would put the third derivative into `d3`. `next` is equivalent to `powers($S_2) coeffs($S_4)` (see Saved Results below).

`curvature` calculates the scaled curvature of the fitted FP function. This is defined as the ratio

$$\frac{\frac{d^2Y}{dX^2}}{K\left[1+\left(\frac{1}{K}\frac{dY}{dX}\right)^2\right]^{\frac{3}{2}}}$$

where $K=(Y_{\max}-Y_{\min})/(X_{\max}-X_{\min})$ is the ratio of the range of fitted Y to the range of X and ensures that the curvature is meaningful (independent of the scales of X and Y).

`dcurvature` is the first derivative of the `curvature` with respect to X.

Saved Results

fpderiv saves in the $S_# macros as follows.

$S_1	degree of FP function comprising first derivative
$S_2	powers of FP function comprising first derivative
$S_3	powers of original (input) FP function
$S_4	coefficients of the derivative of the FP function

Note that S_4 is a matrix with 1 row and $S_1 columns.

References

Royston, P. and D. G. Altman. 1994a. sg26: Using fractional polynomials to model curved regression relationships. *Stata Technical Bulletin* 21: 11–23.

——. 1994b. sg26.1: Fractional polynomials: correction. *Stata Technical Bulletin* 22: 11.

sg32.1	Variance inflation factors and variance-decomposition proportions: Correction

James W. Hardin, Stata Corporation, FAX 409-696-4601, EMAIL stata@stata.com

I have discovered an error in the parsing routines for the vif and colldiag programs. This error occurs when the list of independent variable names exceeds 80 characters. I have fixed the error, and corrected versions of these programs are available on the STB-25 distribution diskette. If you have used these commands, and you had more than 80 characters worth of independent variable names, you should redo the analysis with the new files.

sg35	Robust tests for the equality of variances

Mario A. Cleves, Arkansas Foundation for Medical Care, FAX 501-785-3460

Both the traditional F test for the homogeneity of variances and Bartlett's generalization of this test to K samples are very sensitive to the assumption that the data are drawn from an underlying Gaussian distribution. Levene (1960) proposed a test statistic for equality of variance that was found to be robust under non-normality. Subsequently Brown and Forsythe (1974) proposed alternative formulations of Levene's test statistic using more robust estimators of central tendency in place of the mean. These reformulations were demonstrated to be more robust than Levene's test when dealing with skewed populations.

This insert presents **robvar**, a program that calculates Levene's original statistic along with two reformulations by Brown and Forsythe to provide robust tests for the equality of variances. The syntax for the **robvar** command is

robvar *varname* [**if** *exp*] [**in** *range*] **,** **by**(*groupvar*)

The program displays Levene's statistic (W_0) and two statistics proposed by Brown and Forsythe that replace the mean in Levene's formula with alternative location estimators. The first alternative (W_{50}) replaces the mean with the median. The second alternative replaces the mean with the 10 percent trimmed mean (W_{10}).

Example

You wish to test whether the standard deviation of the length of stay for patients hospitalized for a given medical procedure differs by sex. Your data consists of observations of the length of stay for 1778 patients, 884 males and 894 females.

```
. describe

Contains data from C:\STATA\ROBVAR.DTA
  Obs:  1778 (max= 19723)
 Vars:     2 (max=    99)
Width:     8 (max=   200)
  1. lgthstay     float  %9.0g              LENGTH OF STAY
  2. sex          float  %9.0g              0:MALE 1:FEMALE
Sorted by:
```

Stata's `sdtest` reports the classical test for the equality of variances.

```
. sdtest lgthstay, by(sex)

  Variable |      Obs        Mean   Std. Dev.
  ---------+---------------------------------
         0 |      884    9.087443    9.788475
         1 |      894    8.800671    9.108148
  ---------+---------------------------------
  combined |     1778           .    9.452518

           Ho:      sd(x) = sd(y)    (two-sided test)
               F(883,893) = 1.15
              2*(Pr > F) = 0.0319
```

This test indicates that the null hypothesis that the estimated standard deviations are equal can be rejected at the 5 percent level ($p = .0319$). However, the robust tests reported by `robvar` do not support the rejection of the null hypothesis.

```
. robvar lgthstay, by(sex)

      0:MALE|      Summary of LENGTH OF STAY
    1:FEMALE|        Mean   Std. Dev.       Freq.
------------+-----------------------------------
          0 |   9.0874434   9.7884747         884
          1 |    8.800671   9.1081478         894
------------+-----------------------------------
      Total |   8.9432508   9.4509466        1778

W0=   .5548802   df(1, 1776)     Pr > F = .45642903

W50=  .42704469   df(1, 1776)     Pr > F = .51352721

W10=  .44566503   df(1, 1776)     Pr > F = .50448751
```

The difference between the results of the classical and the robust tests can be traced to the non-normal distribution of the length of stays. Figures 1 and 2 reveal the extent of the skewness of this distribution by sex.

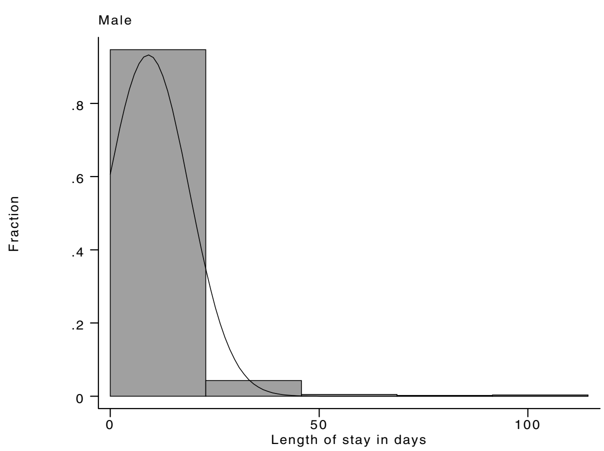

Figure 1

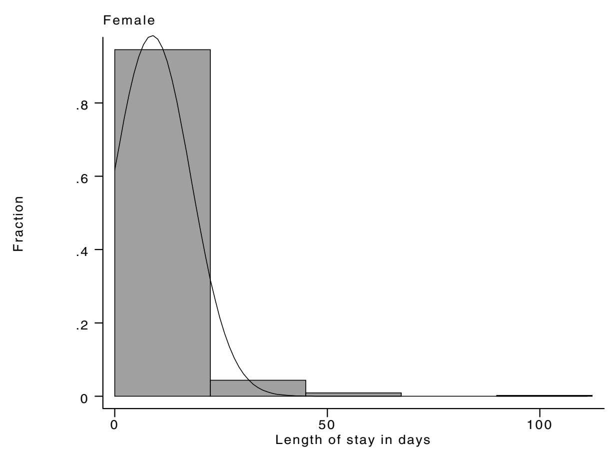

Figure 2

Methods and Formulas

Let X_{ij} be the jth observation of X for the ith group. Let $Z_{ij} = |X_{ij} - \overline{X}_i|$ where $\overline{X}_i$ is the mean of X in the ith group. Levene's test statistic is

$$W_0 = \frac{\sum_i n_i(\overline{Z}_i - \overline{Z})^2/(g-1)}{\sum_i \sum_j (Z_{ij} - \overline{Z}_i)^2/\sum_i(n_i - 1)}$$

where n_i is the number of observations in group i and g is the number of groups. W_{50} is obtained by replacing $\overline{X}_i$ with the ith group median of X_{ij}, while W_{10} is obtained by replacing $\overline{X}_i$ with the 10 percent trimmed mean for group i.

References

Levene, H. 1960. Robust tests for equality of variances. In *Contributions to Probability and Statistics* ed. I. Olkin, 278–292. Palo Alto, CA: Stanford University Press.

Brown, M. B. and A. B. Forsythe. 1974. Robust test for the equality of variances. *Journal of the American Statistical Association* 69: 364–367.

sg36 Tabulating the counts of multiple categorical variables

Peter Sasieni, Imperial Cancer Research Fund, London, FAX (011)-44-171-269-3429

This insert describes `tabw`. For each variable in a list, `tabw` tabulates the number of times it takes on the values 0,1,. . .,9, the number of times it is missing, and the number of times it is equal to some other value. The variables are listed one after the other, so that if there are K (non-string) variables in the list, `tabw` will produce a $K \times 12$ table. String variables do not cause an error message, but are listed separately below the table. The syntax of `tabw` is

tabw *varlist* [**if** *exp*] [**in** *range*]

`tabw` is best understood through examples.

Example 1

```
. describe

Contains data
  Obs:123456 (max=145022)
 Vars:     9 (max=    11)
Width:    16 (max=    24)
  1. sc           byte   %8.0g             Social class
  2. case         byte   %8.0g    case
  3. eight        byte   %8.0g
  4. name         str2   %9s
  5. real         float  %9.0g
  6. month        byte   %8.0g    month
  7. make         str4   %9s
  8. sc_dad       byte   %8.0g             Father´s S.C.
  9. freq         byte   %8.0g
Sorted by:
Note:  Data has changed since last save

. tabw _all in 1/1000
Variable|    0     1     2     3     4     5     6     7     8     9 ****     .
--------+-----------------------------------------------------------------------
sc      |    0   196   199   194   195   196     0     0     0     0    0    20
case    |  616   384     0     0     0     0     0     0     0     0    0     0
eight   |  772     0     0     0     0     0     0     0   196     0    0    32
real    |    0     0     0     0     0     0     0     0     0     0  914    86
month   |    0    83    84    84    84    84    83    83    83    83  249     0
sc_dad  |    0   143   337   335   148    18     0     0     0     0    0    19
freq    |  737   174    49    23    13     2     2     0     0     0    0     0
String variable(s): - name, make
```

Note that due to formatting limitations the maximum number in any column is 99999 (9999 for the column labelled ****). The current version of `tabw` replaces the actual count by the maximum number that can be displayed whenever the actual count exceeds the maximum. A warning is displayed whenever there is the possibility that this has happened.

Example 2

```
. tabw case month freq in 1/10000
WARNING: 9999 in the column labelled **** means at least 9999 "other"
         observations.

Variable|    0     1     2     3     4     5     6     7     8     9 ****     .
--------+-----------------------------------------------------------------------
case    | 5990  4010     0     0     0     0     0     0     0     0    0     0
month   |    0   833   834   834   834   834   833   833   833   833 2499     0
freq    | 7421  1619   603   221    78    39    13     3     2     1    0     0
```

Example 3

```
. tabw name case sc sc_dad
WARNING: 9999 in the column labelled **** means at least 9999 "other"
         observations, similarly 99999 in any other column means
         at least 99999 such observations.

Variable|    0     1     2     3     4     5     6     7     8     9 ****     .
--------+-----------------------------------------------------------------------
case    |74087 49369     0     0     0     0     0     0     0     0    0     0
sc      |    0 24431 24417 24431 24425 24453     0     0     0     0    0  1299
sc_dad  |    0 16951 42181 42163 16763  2580     0     0     0     0    2  2816
String variable(s): - name
```

Note that `tabw` can be quite slow particularly in large data sets. Using `in` instead of `if` will make it run faster when only a selection of the data set is to be tabulated.

Example 4

Finally, here is an example you can replicate using the automobile data supplied with Stata. This example may answer some of your questions about the treatment of floating-point numbers.

```
. use auto
(1978 Automobile Data)

. tabw foreign hdroom make rep78
Variable|    0     1     2     3     4     5     6     7     8     9 ****     .
--------+-----------------------------------------------------------------------
foreign |   52    22     0     0     0     0     0     0     0     0    0     0
hdroom  |    0     0    13    13    10     1     0     0     0     0   37     0
rep78   |    0     2     8    30    18    11     0     0     0     0    0     5
String variable(s):- make

. tabulate rep78
      Repair|
 Record 1978|      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |          2        2.90        2.90
          2 |          8       11.59       14.49
          3 |         30       43.48       57.97
          4 |         18       26.09       84.06
          5 |         11       15.94      100.00
------------+-----------------------------------
      Total |         69      100.00

. tabulate hdroom
    Headroom|
       (in.)|      Freq.     Percent        Cum.
------------+-----------------------------------
        1.5 |          4        5.41        5.41
        2.0 |         13       17.57       22.97
        2.5 |         14       18.92       41.89
        3.0 |         13       17.57       59.46
        3.5 |         15       20.27       79.73
        4.0 |         10       13.51       93.24
        4.5 |          4        5.41       98.65
        5.0 |          1        1.35      100.00
------------+-----------------------------------
      Total |         74      100.00
```

Possible extensions to `tabw`

At the moment there are no options. Possible options include

(i) a `values(#,...,#)` option to specify the list of numbers to be included in the table.

(ii) `nomissing` to exclude the column counting missing values.

(iii) `noother` to exclude the column counting "other" values.

(iv) `format(#,...,#)` where the numbers are the integer width of the columns for each of the values counted (including "other" and "missing"). The default is currently (5,5,5,5,5,5,5,5,5,5,4,5). If just one number was provided the same width would be used for all values.

(v) `percent` to give the percentage of observations equal to each value, instead of the count.

(vi) `round(#)` to specify the amount of rounding accepted in testing to see if the value is approximately equal to the column heading. Thus the program would "count if $7 - \# \leq$ var & var $< 7 + \#$" to see how many values are approximately equal to 7. The default value is 0. For example, using `round(.5)` would count the number of values in the interval [6.5,7.5).

(vii) `recode(#,...,#)` works like Stata's `recode` function except no new variable is generated. The recoded values are tabulated.

sg37	Orthogonal polynomials

William M. Sribney, Stata Corporation, FAX 409-696-4601, EMAIL stata@stata.com

`orthpoly` computes orthogonal polynomials for a variable *varname*. The syntax of `orthpoly` is

`orthpoly` *varname* [*weight*] [`if` *exp*] [`in` *range*] `,` { `generate(`*varlist*`)` `poly(`*matname*`)` } [`degree(`#`)`]

Options

Note: Either one of `generate()` or `poly()` or both must be specified.

`degree(#)` specifies the highest degree polynomial to include. Orthogonal polynomials of degree 1, 2, ..., $d = \#$ are computed. Default is $d = 1$.

`generate(`*varlist*`)` creates d new variables (of type `double`) containing orthogonal polynomials of degree 1, 2, ..., d evaluated at *varname*. The *varlist* must either contain exactly d new variable names or be abbreviated using the styles *newvar*`1-`*newvar* d or *newvar*`*`. For both styles of abbreviation, new variables *newvar*`1`, *newvar*`2`, ..., *newvar* d are generated.

`poly(`*matname*`)` creates a $(d+1) \times (d+1)$ matrix called *matname* containing the coefficients of the orthogonal polynomials. The orthogonal polynomial of degree $i \leq d$ is

matname`[`i`,` $d+1$`]` + *matname*`[`i`, 1]*`*varname* + *matname*`[`i`, 2]*`*varname*`^2`
+ $\cdots$ + *matname*`[`i`,` i`]*`*varname*`^`i

Note that the coefficients corresponding to the constant term are placed in the last column of the matrix. (The rationale for this arrangement is shown in the example below.)

Remarks

When fitting polynomial terms in a regression, orthogonal polynomials are often recommended for two reasons. The first is numerical accuracy. The natural polynomials 1, x, x^2, x^3, ... are highly collinear, and including several terms in a model would create problems for an unsophisticated regression routine. Stata's `regress` command, however, can face a large amount of collinearity and still produce accurate results. Stata users are likely to find orthogonal polynomials useful for the second reason: ease of interpreting results. When orthogonal polynomials are used, $\mathbf{X}'\mathbf{X}$ is diagonal, partial sums of squares become the same as sequential sums of squares, and significance tests are orthogonal.

Examples

Illustrations of syntax:

```
. orthpoly weight, deg(4) generate(pw1 pw2 pw3 pw4)
. orthpoly weight, deg(4) generate(pw1-pw4)
. orthpoly weight, deg(4) generate(pw*)
. orthpoly weight, deg(4) poly(P)
. orthpoly weight, deg(4) gen(pw1-pw4) poly(P)
```

Suppose we wish to fit the model

$$\texttt{mpg} = \beta_0 + \beta_1\,\texttt{weight} + \beta_2\,\texttt{weight}^2 + \beta_3\,\texttt{weight}^3 + \beta_4\,\texttt{weight}^4 + \epsilon$$

We will first compute the regression with natural polynomials, and then do it with orthogonal polynomials.

```
. use auto
(1978 Automobile Data)

. gen double w1 = weight

. gen double w2 = w1*w1

. gen double w3 = w2*w1

. gen double w4 = w3*w1

. regress mpg w1-w4
  Source |       SS       df       MS                  Number of obs =      74
---------+------------------------------               F(  4,    69) =   36.06
   Model |  1652.73666     4  413.184164               Prob > F      =  0.0000
Residual |  790.722803    69  11.4597508               R-squared     =  0.6764
---------+------------------------------               Adj R-squared =  0.6576
   Total |  2443.45946    73  33.4720474               Root MSE      =  3.3852

------------------------------------------------------------------------------
     mpg |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
      w1 |   .0289302   .1161939      0.249   0.804      -.2028704    .2607307
      w2 |  -.0000229   .0000566     -0.404   0.687      -.0001359    .0000901
      w3 |   5.74e-09   1.19e-08      0.482   0.631      -1.80e-08    2.95e-08
      w4 |  -4.86e-13   9.14e-13     -0.532   0.596      -2.31e-12    1.34e-12
   _cons |   23.94421   86.60667      0.276   0.783      -148.8314    196.7198
------------------------------------------------------------------------------

. orthpoly weight, generate(pw*) deg(4)

. regress mpg pw1-pw4
  Source |       SS       df       MS                  Number of obs =      74
---------+------------------------------               F(  4,    69) =   36.06
   Model |  1652.73666     4  413.184164               Prob > F      =  0.0000
Residual |  790.722803    69  11.4597508               R-squared     =  0.6764
---------+------------------------------               Adj R-squared =  0.6576

   Total |  2443.45946    73  33.4720474               Root MSE      =  3.3852

------------------------------------------------------------------------------
     mpg |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     pw1 |  -4.638252   .3935245    -11.786   0.000      -5.423312   -3.853192
     pw2 |   .8263545   .3935245      2.100   0.039       .0412947    1.611414
     pw3 |  -.3068616   .3935245     -0.780   0.438      -1.091921    .4781982
     pw4 |   -.209457   .3935245     -0.532   0.596      -.9945168    .5756027
   _cons |    21.2973   .3935245     54.119   0.000       20.51224    22.08236
------------------------------------------------------------------------------

. orthpoly weight, poly(P) deg(4)

. matrix bp = get(_b)

. matrix b = bp*P

. matrix list b

b[1,5]
          deg1        deg2        deg3        deg4       _cons
y1   .02893016  -.00002291   5.745e-09  -4.862e-13   23.944212
```

Compare the P-values of the terms in the natural-polynomial regression to those in the orthogonal-polynomial regression. With orthogonal polynomials, it is easy to see that the cubic and quartic terms are nonsignificant and that the constant, linear, and quadratic terms each have $P < 0.05$.

The example also illustrates how the matrix `P` obtained with the `poly()` option can be used to transform coefficients for orthogonal polynomials to coefficients for natural polynomials. The row vector `bp` contains the coefficients from the orthogonal-polynomial regression; `matrix b = bp*P` transforms them to coefficients of natural polynomials. These are, as they should be, the same as the coefficients from the natural-polynomial regression.

Methods and Formulas

`orthpoly` uses the Christoffel–Darboux recurrence formula. They are normalized so that $\mathbf{X}'\mathbf{D}\mathbf{X} = N\mathbf{I}$, where $\mathbf{D} = \mathrm{diag}(w_1, w_2, \ldots, w_n)$ with w_1, w_2, $\ldots$, w_n the weights (all 1 if weights not specified) and $N = \sum_{i=1}^{n} w_i$. (If the weights are `aweights`, they are first normalized to sum to the number of observations.)

Reference

Abramowitz, M. and I. A. Stegun, eds. 1968. *Handbook of Mathematical Functions*, 7th printing. Washington, D.C.: National Bureau of Standards.

sg38	Generating quantiles

William M. Sribney, Stata Corporation, FAX 409-696-4601, EMAIL stata@stata.com

The syntax of `mkquant` is

`mkquant` *varname* [*weight*] [`if` *exp*] [`in` *range*] , `genq(`*newvar1*`)` [`genp(`*newvar2*`)` `number(`*#*`)`]

`aweight`s and `fweight`s are allowed.

`mkquant` computes the pth quantiles of *varname* for $p = i/n$ with $i = 0, 1, \ldots, n$. The $p = 0$ quantile is defined as the minimum of *varname* and the $p = 1$ quantile as the maximum. (The pth quantile is, of course, identical to the $100\,p$th percentile.)

Options

`genq(`*newvar1*`)` is not optional. The generated quantiles are stored in the new variable *newvar1*. If *varname* is of type `double`, then so is *newvar1*; otherwise, *newvar1* is a `float`.

`genp(`*newvar2*`)`, if specified, generates the corresponding empirical probabilities $p = i/n$ for $i = 0, 1, \ldots, n$ and stores them in the new variable *newvar2* (of type `float`).

`number(`#`)` specifies the number of quantiles $n = \#$. Default is $n = 100$.

Remarks

The Stata commands `centile` and `egen` (with the `pctile` function) will calculate any specified percentile. But computing and storing a large number of percentiles with these commands is somewhat cumbersome. Furthermore, neither of these commands allow weights.

`mkquant` will quickly compute a large number of quantiles and place them in a new variable sequentially; i.e., the $p = i/n$ quantile is stored in observation $i + 1$. This storage scheme allows one to compute quantiles (with the same n) for additional variables and to have these quantiles match up with those already calculated. The quantiles can then be directly compared.

Example

```
. mkquant x1, genq(q1) genp(p) n(20)
. mkquant x2, genq(q2) n(20)
. list p q1 q2 in 1/21
              p         q1         q2
  1.          0          5          2
  2.        .05         15         14
  3.         .1         16         16
  4.        .15         17         17
  5.         .2         18         19
  6.        .25         19         20
  7.         .3         19       20.5
  8.        .35         20         21
  9.         .4         20         22
 10.        .45         21         23
 11.         .5         22         24
 12.        .55         22         24
 13.         .6         23         25
 14.        .65         23         26
 15.         .7         24         26
 16.        .75         24         27
 17.         .8         25         28
 18.        .85         26         30
 19.         .9         27         31
 20.        .95         29         33
 21.          1         38         42
```

Methods and Formulas

Let the values of x sorted in ascending order be x_1, x_2, ..., x_N, with corresponding weights w_1, w_2, ..., w_N (all equal to 1 if weights not specified). Let

$$W_i = \frac{\sum_{j=1}^{i} w_j}{\sum_{j=1}^{N} w_j}$$

Then the p th quantile of x for $0 < p < 1$ is

$$q(p) = \begin{cases} x_i & \text{if } W_{i-1} < p < W_i \\ (x_i + x_{i+1})/2 & \text{if } p = W_i \end{cases}$$

We define $q(0) = x_1$ and $q(1) = x_N$.

sg39	Independent percentages in tables

Benjamin Miller, Department of Psychology, Simmons College, EMAIL bmiller@vmsvax.simmons.edu

Stata provides options for computing percentages in tables by row, by column, or by cell. But, for answering certain kinds of questions, these options can be awkward.

Consider a survey of college and university faculty that contains variables that identify the gender of the faculty member surveyed, whether they are tenured, the geographical region of their institution, and whether the institution is public or private. We can easily find, for example, the percentage of tenured faculty at public institutions who are women.

```
. use tenure

. tabulate private gender if tenure, row

           | Gender
   Pub/Priv|    female       male |     Total
-----------+----------------------+----------
    public |        35        152 |       187
           |     18.72      81.28 |    100.00
-----------+----------------------+----------
   private |        63        210 |       273
           |     23.08      76.92 |    100.00
-----------+----------------------+----------
      Total|        98        362 |       460
           |     21.30      78.70 |    100.00
```

As the `tabulate` command shows, 35 (19 percent) of the 187 tenured, public institution faculty in these data are women.

Similarly, we can find the percentage of tenured women who are at public institutions.

```
. tabulate private gender if tenure, column

           | Gender
   Pub/Priv|    female       male |     Total
-----------+----------------------+----------
    public |        35        152 |       187
           |     35.71      41.99 |     40.65
-----------+----------------------+----------
   private |        63        210 |       273
           |     64.29      58.01 |     59.35
-----------+----------------------+----------
      Total|        98        362 |       460
           |    100.00     100.00 |    100.00
```

The same 35 women now constitute 36 percent of the 98 female faculty members in the survey.

If we want to know what percentage of women are tenured at both public and private institutions, we can type

```
. sort private
```

```
. by private: tabulate gender tenure, row

-> private=  public
           | Is tenured?
     Gender|        no        yes |     Total
-----------+----------------------+----------
    female |        70         35 |       105
           |     66.67      33.33 |    100.00
-----------+----------------------+----------
      male |        52        152 |       204
           |     25.49      74.51 |    100.00
-----------+----------------------+----------
      Total|       122        187 |       309
           |     39.48      60.52 |    100.00

-> private= private
           | Is tenured?
     Gender|        no        yes |     Total
-----------+----------------------+----------
    female |        94         63 |       157
           |     59.87      40.13 |    100.00
-----------+----------------------+----------
      male |        72        210 |       282
           |     25.53      74.47 |    100.00
-----------+----------------------+----------
      Total|       166        273 |       439
           |     37.81      62.19 |    100.00
```

In this sample, 40 percent of the female faculty at private institutions are tenured compared to 33 percent of the female faculty at public institutions.

This method is satisfactory, but not ideal. One problem is that the percentages in one column add no information to those in the other column, creating unnecessary bulk. This problem can be avoided by directly computing the percentage of cases in each cell who are tenured. I have created `iptab` for this purpose:

```
. iptab private gender tenure if tenure
                     Means and Frequencies of __000004

           | Gender
   Pub/Priv|    female       male      Total
-----------+----------------------+----------
    public |      33.3       74.5 |      66.8
           |        35        152 |       187
-----------+----------------------+----------
   private |      40.1       74.5 |      66.5
           |        63        210 |       273
-----------+----------------------+----------
     Total |      37.7       74.5 |      66.6
           |        98        362 |       460
```

Now the percentages are all independent of one another. (The mysterious title of this table will be clear in a moment.)

The other problem with the `by` *variable*`: tabulate` method is that the amount of output can be cumbersome when the `by`-variable takes many values. For example, we might examine the gender/tenure relation by geographical region:

```
. iptab region gender tenure if tenure
                     Means and Frequencies of __00000D

           | Gender
     Region|    female       male      Total
-----------+----------------------+----------
    N-East |      37.0       75.6 |      66.6
           |        30         99 |       129
-----------+----------------------+----------
    S-East |      44.4       72.2 |      65.2
           |        28         83 |       111
-----------+----------------------+----------
   Midwest |      35.4       77.7 |      69.8
           |        23        101 |       124
-----------+----------------------+----------
```

```
   Mountain |      31.3       57.9 |      49.6
            |         5         11 |        16
-----------+----------------------+----------
     S-West |      25.0       78.6 |      72.8
            |         4         33 |        37
-----------+----------------------+----------
    W-Coast |      38.1       71.4 |      65.2
            |         8         35 |        43
-----------+----------------------+----------
      Total |      38.1       74.7 |      66.9
            |        98        362 |       460
```

`iptab` produces a single table that allows immediate comparison by row and by column. The alternative to `iptab` (`by region: tabulate`) would produce six two-by-two tables.

Two additional conveniences provided by `iptab` have to do with

(i) categorical dependent variables with more than two values or continuous variables where we are interested in a particular value or range; and

(ii) missing values in the dependent variable.

Here is an example that illustrates both conveniences.

We administered a two-item questionnaire to a sample of men and women, asking them to rate each item on a scale of 1 (low) to 5 (high). Some respondents did not rate both items, so there are missing values on rating. To assess gender differences in responses to the questionnaire, we might ask what percentage of each gender gave a given item a high rating (say, 4 or 5). We can do this with the "`by:`" method as follows:

```
. use ratings, clear

. *
. *       dichotomize the dependent variable
. *
. generate ratehigh=rating > 3

. replace ratehigh=. if rating==.
(43 real changes made, 43 to missing)

. sort item

. by item: tabulate gender ratehigh, row

-> item=  item 1
  gender of| ratehigh
 respondent|         0          1 |     Total
-----------+----------------------+----------
     female |        74        184 |       258
            |     28.68      71.32 |    100.00
-----------+----------------------+----------
       male |       187        287 |       474
            |     39.45      60.55 |    100.00
-----------+----------------------+----------
       Total|       261        471 |       732
            |     35.66      64.34 |    100.00


-> item=  item 2
  gender of| ratehigh
 respondent|         0          1 |     Total
-----------+----------------------+----------
     female |        37        221 |       258
            |     14.34      85.66 |    100.00
-----------+----------------------+----------
       male |        76        399 |       475
            |     16.00      84.00 |    100.00
-----------+----------------------+----------
       Total|       113        620 |       733
            |     15.42      84.58 |    100.00
```

Alternatively, we can use `iptab`, which allows us to select the dependent variable value(s) of interest without creating a temporary variable (`ratehigh`) and which removes missing values automatically.

```
. iptab item gender rating if rating>3
                     Means and Frequencies of __00000M
items to be| gender of respondent
      rated|    female       male      Total
-----------+----------------------+----------
     item 1 |      71.3       60.5 |      64.8
            |       184        287 |       471
-----------+----------------------+----------
     item 2 |      85.7       84.0 |      84.6
            |       221        399 |       620
-----------+----------------------+----------
      Total |      79.1       74.2 |      76.0
            |       405        686 |      1091
```

iptab's internal operations are inelegant but effective. The numerator of the proportion in each cell is the number of observations that have the target values of rating, in this case 4 or 5. This can be calculated as follows:

```
. generate target = rating>3 & rating~=.
. egen numer = sum(target), by(item gender)
```

The denominator is the number of observations in each cell with nonmissing values on all relevant variables.

```
. generate nonmiss = (item~=. & gender~=. & rating~=.)
. egen denom = sum(nonmiss), by(item gender)
```

Now the percentages are computed and displayed.

```
. generate percent=(numer/denom)*100
(14 missing values generated)
. format percent %4.1f
. tabulate item gender if target, sum(percent) nostandard
                     Means and Frequencies of percent
items to be| gender of respondent
      rated|    female       male      Total
-----------+----------------------+----------
     item 1 |      71.3       60.5 |      64.8
            |       184        287 |       471
-----------+----------------------+----------
     item 2 |      85.7       84.0 |      84.6
            |       221        399 |       620
-----------+----------------------+----------
      Total |      79.1       74.2 |      76.0
            |       405        686 |      1091
```

The inelegance of the solution lies in the redundant computation and storage that this method entails. The numerator and denominator for a given cell are recorded in each target observation for that combination of item and group; once would be enough. By the same token, the percent is computed and stored redundantly. However, this approach allows the use of `tabulate, summarize()` to construct and display the table. Because `summarize` is in fact finding the means of sets of identical values, there is no point in displaying the standard deviations, all of which are zero.

The cell frequencies are, in fact, the numbers of target observations. The marginal percentages are weighted means of the cell percentages. The tables' mysterious titles reflect the fact that `iptab` puts the percentages (and everything else) in temporary macros; in any case the percentages are not really the means of anything.

Syntax

`iptab` *row_var column_var dep_var* `if` *target_exp* [`,` `nofreq` `wrap`]

The *target_exp* in the `if` clause is of the form *dep_var* $==x$, *dep_var* $<x$, etc. Note that the `if` clause is required. The order of the row and column variables does not matter—switching item and group in the example simply transposes the rows and columns.

snp8 | Robust scatterplot smoothing: enhancements to Stata's ksm

Isaías Hazarmabeth Salgado-Ugarte and Makoto Shimizu, University of Tokyo,
Fac. of Agriculture, Dept. of Fisheries, Japan FAX (011)-81-3-3812-0529, EMAIL isalgado@tansei.cc.u-tokyo.ac.jp

Stata's `ksm` calculates weighted and unweighted scatterplot smooths. Given data on a pair of variables, x_i and y_i, where the index i is defined such that $i < i' \Rightarrow x_i \le x_{i'}$, `ksm` calculates smoothed values $\widehat{y}_i$ that are conditioned on values of x

close to x_i. This is roughly equivalent to drawing a scatterplot of y_i versus x_i, passing a smooth curve through the points, and recording the y-values of the curve for each value of x_i. This procedure is motivated by the assumption that

$$y_i = f(x_i) + \epsilon_i$$

where $f()$ is an unspecified smooth function and ϵ is a random disturbance from an unknown distribution.

`ksm` is a valuable tool both for exploratory data analysis and for use in nonparametric and semiparametric estimation techniques. However, as implemented, `ksm` presents two limitations. First, the algorithm used by `ksm` uses fewer (x_i, y_i) pairs to estimate the endpoints of the smooth than are used in the body of the smooth. This feature makes `ksm` more "local" in the endpoints and, thus, more likely to be influenced by one or two discrepant values in the vicinity of the endpoints. Second, the `lowess` option of `ksm` does not implement the robustness weights recommended by Cleveland (1979) in his initial presentation of the lowess scatterplot smoother.

This insert offers two enhancements to `ksm` that overcome these limitations. The first enhancement is `adjksm`, a modified version of `ksm` that holds the bandwidth of the smoother constant across the range of x-values. The second enhancement consists of two programs that calculate a lowess smooth using Cleveland's robust weights. `lorobwei` calculates the weights recommended by Cleveland, while `roblowes` uses these weights to compute the lowess smooth.

Holding bandwidth constant across the x-axis

As implemented, the bandwidth of `ksm` is not constant across the domain of x values. In particular, fewer points are used to calculate the smooth at the endpoints than in the middle of the domain. For instance, the table below displays the number of points included when a smooth is calculated for a data set with ten observations.

`_n`	number of observations
1	3
2	4
3	5
4	5
5	5
6	5
7	5
8	5
9	4
10	3

`ksm` calculates the smoothed values in observations three through eight based on half the sample. The smoothed values in observations two and nine are based on 40 percent of the sample, and the smoothed endpoints are based on only 30 percent of the sample. As a consequence, the endpoints of the smooth are likely to track the behavior of the actual y-values more closely than the endpoint smooth values from Cleveland's (1979) algorithm.

We have modified `ksm` to hold the bandwidth constant across the domain of x. Our modified program is called `adjksm`. Other than this modification, `adjksm` shares all the characteristics of the original `ksm`.

Using Cleveland's robustness weights

An important component of the locally weighted scatterplot smoother (*lowess*) proposed by Cleveland is a set of robustness weights that protects the iterative smoothing process from the influence of discrepant y-values. These weights are not implemented in `ksm`, even when the `lowess` option is specified.

We have made a separate set of adjustments to `ksm` to implement Cleveland's robustness weights. We regard these adjustments as a temporary expedient, until such time as Stata implements Cleveland's robust weights in `ksm`.

Our programs calculate each iteration of Cleveland's version of lowess in two steps. The first step, calculating the robust weights, is performed by our program `lorobwei`, which is a mnemonic for locally robust weights. The syntax for `lorobwei` is

`lorobwei` *y yksm*

where *y* is the y-variable in the scatterplot smooth and *yksm* is the smooth calculated by `adjksm`. In other words, *yksm* is an input to the `lorobwei` procedure.

`lorobwei` generates a new variable named `robwei` containing robust weights calculated from the bisquare function, that is, Tukey's biweight function as described in Mosteller and Tukey (1977), Cleveland (1979), Chambers et al. (1983) or Goodall (1983, 1990). The values in `robwei` are used in the second step by the program `roblowes` which estimates the lowess smooth. The syntax of `roblowess` is

`roblowes` *y xvar bwidth*

where *bwidth* is a number between 0 and 1 that specifies the desired bandwidth as a fraction of the sample. `roblowes` generates a new variable, `lowerob`, that contains the lowess smooth.

The lowess smooth is calculated by iterating over these two steps. Note that the variable `robwei` must be dropped before each call to the program `lorobwei`, and the variable `lowerob` must be dropped before each call to the program `roblowes`.

Example

This example is adapted from the original description of lowess in Cleveland. The example uses the well-known abrasion loss data (called the "Rubber specimen data" in Chambers et al.).

We begin by comparing the performance of Stata's `ksm` to our `adjksm` in smoothing abrasion loss against tensile strength.

```
. use dta\rubber
. ksm alr tsr, gen(oldksm) bwidth(.5) nograph
. adjksm alr tsr, gen(newksm) bwidth(.5) nograph
. label variable oldksm "Stata´s ksm"
. label variable newksm "New ksm"
. graph oldksm newksm alr tsr, c(ll.) s(pdO) title(Abrasion loss data)
```
(graph appears, see Figure 1)

The sensitivity of Stata's smooth to endpoint values is clear in this figure.

Now we compare Stata's version of lowess to the full Cleveland procedure using robust weights. We perform two iterations of the lowess procedure.

```
. ksm alr tsr, gen(oldlow) lowess bwidth(.5) nograph
. label variable oldlow "Stata´s lowess"
. adjksm alr tsr, gen(low0) lowess bwidth(.5) nograph
*
* Iteration 1
*
. lorobwei alr low0
. roblowes alr tsr .5
. rename lowerob low1
. rename robwei wei1
*
* Iteration 2
*
. lorobwei alr low1
. roblowes alr tsr .5
. rename lowerob newlow
. label variable newlow "New lowess"
. graph oldlow newlow alr tsr, c(ll.) s(pdO) title(Abrasion loss data)
```
(graph appears, see Figure 2)

This example illustrates the importance of the robustness weights: the outlier in the lower left does not disturb Cleveland's lowess smooth.

We have tested our programs on some additional data sets: the "Hamster Hibernation Data" (`hiber.dta`), the "Graph Areas" (`grafarea.dta`), and the "Made-up data" (`madeup.dta`) from Chambers et al. (1983); and the "Tadpoles Data" (`tadpole.dta`) from Travis (1983) and reanalyzed in Trexler and Travis (1993). We have included these data sets on the distribution diskette, and we encourage readers to experiment with them. If you do, you will note that the robustness weights have a more significant impact on the smooth when the number of observations is small and when there are y-outliers near the end points of the x-values.

References

Chambers, J. M., W. S. Cleveland, B. Kleiner, and P. A. Tukey. 1983. *Graphical Methods for Data Analysis* Belmont, CA: Wadsworth.

Cleveland, W. S. 1979. Robust locally-weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* 74: 829–836.

Goodall, C. 1983. M-estimators of location: an outline of the theory. In *Understanding Robust and Exploratory Data Analysis* ed. D. C. Hoaglin, F. Mosteller, and J. W. Tukey, 339–403. New York: John Wiley & Sons.

——. 1990. A survey of smoothing techniques. In *Modern Methods of Data Analysis* ed. J. Fox and J. S. Long, 126–176. Newbury Park, CA: Sage Publications.

Mosteller, F. and J. W. Tukey. 1977. *Data Analysis and Regression* Reading, MA: Addison–Wesley.

Travis, J. 1983. Variation in growth and survival of Hyla gratiosa larvae in experimental enclosures. *Copeia* 1983: 232–237.

Trexler, J. C. and J. Travis. 1993. Nontraditional regression analyses. *Ecology* 74: 1629–1637.

Figures

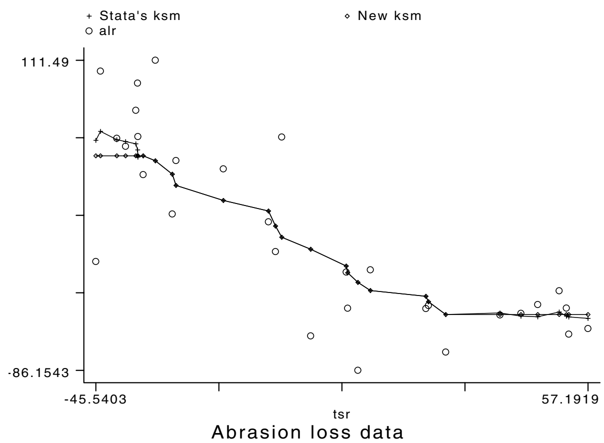

Figure 1

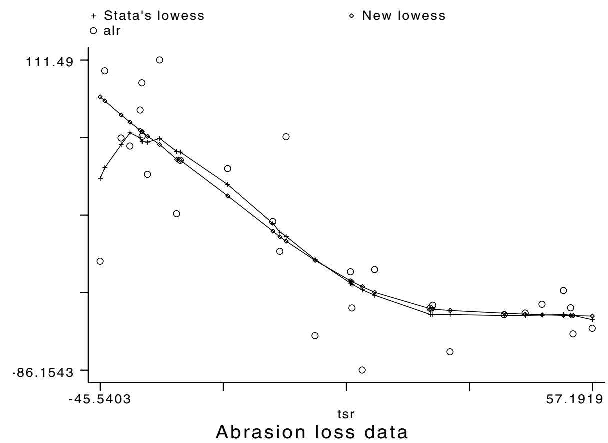

Figure 2

sts10	Prais–Winsten regression

James W. Hardin, Stata Corporation, FAX 409-696-4601, EMAIL stata@stata.com

The syntax of `prais` is

`prais` *depvar* [*varlist*] [`in` *range*] [, `level(`*#*`)` `nolog` `iterate(`*#*`)` `tol(`*#*`)`]

`prais` shares the features of all estimation commands; see [4] estimate, but note that to use `predict`, you must first type `gen _iter=1`.

Description

`prais` estimates a linear regression of *depvar* on *varlist* that is corrected for serially correlated residuals using the Prais–Winsten (1954) estimator. This estimator improves on the Cochrane–Orcutt (1949) method in that the first observation is preserved in the estimation routine.

Options

`level(`*#*`)` specifies the significance level for confidence intervals of the coefficients; see [4] estimate.

`nolog` suppresses the iteration log.

`iterate(`*#*`)` specifies the maximum number of iterations and defaults to 100, a number close enough to infinity to be nonbinding. You should never have to specify this option.

`tol(#)` specifies the minimum change in the estimated autocorrelation parameter between iterations before convergence can be declared and defaults to 0.001.

Remarks

The most common autocorrelated error process assumed for the vector $\mathbf{u}$ is the first order autoregressive process. Under this assumption, the linear regression model may be written

$$y_t = \mathbf{x}'\mathcal{B} + u_t$$

where the errors satisfy

$$u_t = \rho\, u_{t-1} + e_t$$

and the e_t are independent and identically distributed as $\mathrm{N}(0,\sigma^2)$. The covariance matrix Ψ of the error term e may then be written as

$$\Psi = \frac{1}{1-\rho^2}\begin{pmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{T-1} \\ \rho & 1 & \rho & \cdots & \rho^{T-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{T-1} & \rho^{T-2} & \rho^{T-3} & \cdots & 1 \end{pmatrix}$$

The inverse of the covariance matrix may then be written as

$$\Psi^{-1} = \begin{pmatrix} \sqrt{1-\rho^2} & 0 & 0 & \cdots & 0 & 0 \\ -\rho & 1 & 0 & \cdots & 0 & 0 \\ 0 & -\rho & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & -\rho & 1 \end{pmatrix}$$

where we use the inverse to define the matrix P such that $P'P = \Psi^{-1}$.

The Prais–Winsten estimator is a generalized least squares (GLS) estimator. The Prais–Winsten method (as described in Judge et al. 1985) is derived from the AR(1) model for the error term described above. Where the Cochrane–Orcutt method uses a lag definition and loses the first observation in the iterative method, the Prais–Winsten method preserves that first observation. In small samples, this can be a significant advantage.

Example

You wish to estimate a time-series model of `usr` on `idle` but are concerned that the residuals may be serially correlated:

```
. corc usr idle, nolog
(Cochrane-Orcutt regression)
  Source |       SS       df       MS                  Number of obs =      29
---------+------------------------------               F(  1,    27) =    6.51
   Model |  40.2374309     1  40.2374309               Prob > F      =  0.0167
Residual |  166.898634    27  6.18143089               R-squared     =  0.1943
---------+------------------------------               Adj R-squared =  0.1644
   Total |  207.136065    28   7.3977166               Root MSE      =  2.4862

------------------------------------------------------------------------------
     usr |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    idle |  -.1256177   .0492357     -2.551   0.017      -.2266411   -.0245944
   _iter |   14.56028    4.27173      3.409   0.002       5.795409    23.32514
------------------------------------------------------------------------------
     rho |     0.5705     0.1541      3.702   0.001         0.2549      0.8861
------------------------------------------------------------------------------
```

The estimated model is

$$\texttt{usr}_t = -.1256\,\texttt{idle}_t + 14.56 + u_t \qquad \text{and} \qquad u_t = .5705\, u_{t-1} + e_t$$

Comparing this to the Prais–Winsten method we see that

```
. prais usr idle, nolog
(Prais-Winsten regression)
  Source |       SS       df       MS                  Number of obs =      30
---------+------------------------------               F(  1,    28) =    4.31
   Model |  27.0976894     1  27.0976894               Prob > F      =  0.0471
Residual |  175.842266    28  6.28008092               R-squared     =  0.1335
---------+------------------------------               Adj R-squared =  0.1026
   Total |  202.939955    29  6.99792949               Root MSE      =   2.506
------------------------------------------------------------------------------
     usr |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
    idle |  -.0761436   .0366564     -2.077   0.047      -.1512309   -.0010564
   _iter |   10.61985   3.485955      3.046   0.005         3.4792    17.76051
------------------------------------------------------------------------------
     rho |     0.6460     0.1424      4.535   0.000         0.3543      0.9378
------------------------------------------------------------------------------
```

where the Prais–Winsten estimated model is

$$\texttt{usr}_t = -.0761\,\texttt{idle}_t + 10.62 + u_t \qquad \text{and} \qquad u_t = .6460\,u_{t-1} + e_t$$

A comparison of the predicted regression line for the Cochrane–Orcutt, Prais–Winsten, and classic OLS for this data looks like

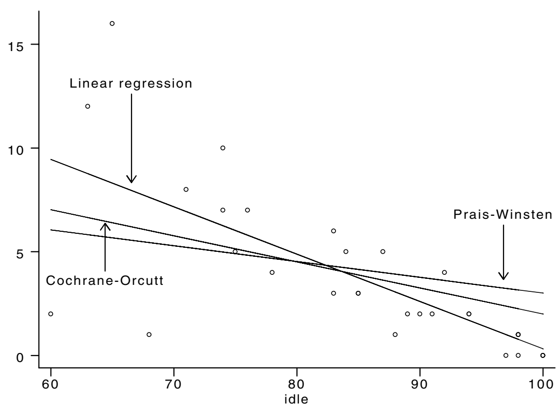

Figure 1

Results

The `prais` command stores the command name in `S_E_cmd` and the name of the dependent variable in `S_E_depv`. In addition, `prais` saves in the macro `S_1` the estimate of rho and in `S_2` its standard error.

Methods and Formulas

Consider the command '`prais` *y x z*'. The 0-th iteration is obtained by estimating a, b, and c from the regression:

$$y_t = ax_t + bz_t + c + u_t$$

The auxiliary regression

$$u_t = ru_{t-1} + e_t$$

is then estimated to obtain an estimate of the correlation in the residuals. Next we estimate equation (1) for $t = 2, \ldots, n$

$$y_t - ry_{t-1} = a(x_t - rx_{t-1}) + b(z_t - rz_{t-1}) + c(1 - r) + v_t \tag{1}$$

and equation $(1')$ for $t = 1$

$$\sqrt{1-r^2}y_1 = a(\sqrt{1-r^2}x_1) + b(\sqrt{1-r^2}z_1) + c\sqrt{1-r^2} + \sqrt{1-r^2}v_1 \tag{1'}$$

Thus, the difference between the Cochrane–Orcutt and the Prais–Winsten methods are that the latter uses equation ($1'$) in addition to equation (1), while the former uses only equation (1) and necessarily decreases the sample size by one.

Equations (1) and ($1'$) are then used to obtain estimates of a, b, and c, and take these estimates to produce

$$\widehat{y} = ax_t + bz_t + c$$

r is estimated from

$$y_t - \widehat{y}_t = r(y_{t-1} - \widehat{y}_{t-1}) + u_t \tag{2}$$

We then re-estimate equation (1) using the new estimate of r, and continue to iterate between (1) and (2) until r converges.

Convergence is declared after `iterate()` iterations or when the absolute difference in the estimated correlation between two iterations is less than `tol()`. Sargan (1964) has shown that this process will always converge.

All reported statistics are based on the r-transformed variables.

References

Cochrane, D. and G. H. Orcutt. 1949. Application of least-squares regression to relationships containing autocorrelated error terms. *Journal of the American Statistical Association* 44: 32–61.

Judge, G. G., W. E. Griffiths, R. C. Hill, H. Lütkepohl, and T. C. Lee. 1985. *The Theory and Practice of Econometrics.* 2d ed. New York: John Wiley & Sons.

Prais, S. J. and C. B. Winsten. 1954. Trend Estimators and Serial Correlation. *Cowles Commission Discussion Paper No. 383*, Chicago.

Sargan, J. D. 1964. Wages and prices in the United Kingdom: a study in econometric methodology. In *Econometric Analysis for National Economic Planning*, ed. P. E. Hart, G. Mills, J. K. Whitaker, 25–64. London: Butterworths.

zz5 Cumulative index for STB-19–STB-24

[an] Announcements

STB-19	2	an42	STB-13–STB-18 available in bound format *S. Becketti*
STB-19	2	an43	New address for STB office *S. Becketti*
STB-19	3	an44	StataQuest: Stata for teaching *S. Loll*
STB-20	2	an44.1	StataQuest disk enclosed (really) *P. Branton*
STB-19	4	an45	Stata and Stage now available for DEC Alpha *T. McGuire*
STB-20	2	an46	Stata and Stage now available for IBM PowerPC *T. McGuire*
STB-21	2	an47	New associate editors *S. Becketti*
STB-21	3	an48	Updated CPS labor extracts available *D. Feenberg*
STB-21	4	an49	Stata listserver available *D. Wormuth*
STB-22	2	an50	Submission guidelines *S. Becketti*
STB-23	2	an51	Call for suggestions *S. Becketti*
STB-23	2	an52	Stata 4.0 released *P. Branton*
STB-23	2	an53	Implications of Stata 4.0 for the STB *S. Becketti*

[crc] CRC-Provided Support Materials

STB-20	2	crc36	Clarification on analytic weights with linear regression
STB-23	3	crc37	Commonly asked questions about Stata for Windows
STB-23	4	crc38	Installing Stata for Windows under OS/2 Warp

[dm] Data Management

STB-19	4	dm17	Conversions for international date formats *P. Ryan*
STB-19	5	dm18	Adding trailing moving averages to the egen command *S. Becketti*
STB-20	3	dm19	Merging raw data and dictionary files *J. Nash*
STB-20	6	dm20	Date functions *A. Riley*
STB-22	3	dm21	Bringing large data sets into memory *R. Farmer*
STB-22	5	dm22	Sorting in descending order *D. Mabb*
STB-22	6	dm23	Saving a subset of the current data set *D. Mabb*
STB-23	8	dm24	Producing formatted tables in Stata *W. Rogers*
STB-24	2	dm25	Recoding in steps *S. Becketti*
STB-24	4	dm26	Labeling graphs with date formats *A. Riley*
STB-24	5	dm27	An improved collapse, with weights *W. Gould*

[dt] Data Sets

STB-22	6	dt2	Reading public use microdata samples into Stata	*C. Sigmund & D. Judson*

[gr] Graphics

STB-19	8	gr14	dotplot: Comparative scatterplots	*P. Sasieni & P. Royston*
STB-19	11	gr15	Incorporating Stata graphs in TEX documents using an HP printer	*S. Becketti*
STB-23	11	gr16	Convex hull programs	*J. Gray & T. McGuire*
STB-24	8	gr17	Switching graphics windows in Unix	*P. Sasieni*

[ip] Instruction on Programming

STB-20	8	ip6	Storing variables in vectors and matrices	*K. Heinecke*
STB-20	10	ip6.1	Data and matrices	*W. Gould*
STB-24	9	ip6.2	Storing matrices as variables	*W. Sribney*
STB-23	16	ip7	A utility for debugging Stata programs	*T. Schmidt*

[os] Operating System, etc.

STB-19	14	os12	Windowed interfaces for Stata	*W. Gould*
STB-19	15	os13	Using awk and fgrep for selective extraction from Stata log files	*N. Cox*
STB-20	10	os14	A program to format raw data files	*P. Swagel*
STB-23	5	os15	The impact of the Pentium FDIV bug on Stata users	*W. Gould*

[sbe] Biostatistics and Epidemiology

STB-21	5	sbe11	Direct Standardization	*T. McGuire*

[sed] Exploratory Data Analysis

STB-21	10	sed8	Finding significant gaps in univariate distributions	*R. Goldstein*
STB-24	10	sed9	Symmetric nearest neighbor linear smoothers	*P. Sasieni*

[sg] General Statistics

STB-24	14	sg21.1	Equivalency testing: Correction	*R. Goldstein*
STB-19	17	sg22.3	Generalized linear models: revision of glm. Rejoinder	*P. Royston*
STB-20	12	sg25	Interaction expansion	*W. Gould*
STB-21	11	sg26	Using fractional polynomials to model curved regression relationships	*P. Royston & D. Altman*
STB-22	11	sg26.1	Fractional polynomials: Correction	*P. Royston & D. Altman*
STB-24	14	sg26.2	Calculating and graphing fractional polynomials	*S. Becketti*
STB-22	12	sg27	The overlapping coefficient and an "improved" rank-sum statistic	*R. Goldstein*
STB-22	15	sg28	Multiple comparisons of categories after regression-like methods	*W. Rogers*
STB-23	18	sg29	Tabulation of observed/expected ratios and confidence intervals	*P. Sasieni*
STB-23	20	sg30	Measures of inequality in Stata	*E. Whitehouse*
STB-23	23	sg31	Measures of diversity: absolute and relative	*R. Goldstein*
STB-24	17	sg32	Variance inflation factors and variance-decomposition proportions	*J. Hardin*
STB-24	22	sg33	Calculation of adjusted means and adjusted proportions	*J. Garrett*
STB-24	25	sg34	Jackknife estimation	*W. Gould*

[snp] Nonparametric methods

STB-22	19	snp7	Natural cubic splines	*P. Sasieni*
STB-24	29	snp7.1	Natural cubic splines: Correction	*P. Sasieni*

[sqv] Analysis of Qualitative Variables

STB-19	17	sqv9	Probit coefficients as changes in probabilities	*W. Gould*
STB-23	26	sqv10	Expanded multinomial comparisons	*W. Rogers*

[ssa] Survival Analysis

STB-19	22	ssa3	Adjusted survival curves *W. Rogers*
STB-19	23	ssa4	Ex post tests and diagnostics for a proportional hazards model *W. Rogers*
STB-19	28	ssa5	Note on time intervals in time-varying Cox regression *W. Rogers*
STB-22	22	ssa6	Utilities for survival analysis with time-varying regressors *P. Bocquier*

[ssi] Simulation and Random Numbers

STB-19	28	ssi5.3	Correction to Ridders' method *T. McGuire*
STB-24	29	ssi5.4	Correction to error term in Ridders' method *A. Riley*
STB-20	18	ssi6	Routines to speed Monte Carlo experiments *W. Gould*
STB-20	22	ssi6.1	Simplified Monte Carlo simulations *W. Gould*
STB-21	24	ssi6.2	Faster and easier bootstrap estimation *W. Gould*

[sss] Social Science and Psychometrics

STB-22	29	sss1.1	Updated U.S. marginal income tax rate function *T. Schmidt*

[sts] Time Series and Econometrics

STB-19	28	sts7.2	A library of time series programs for Stata (Update) *S. Becketti*
STB-20	25	sts7.3	A library of time series programs for Stata (Update) *S. Becketti*
STB-22	29	sts7.4	A library of time series programs for Stata (Update) *S. Becketti*
STB-23	28	sts7.5	A library of time series programs for Stata (Update) *S. Becketti*
STB-24	30	sts7.6	A library of time series programs for Stata (Update) *S. Becketti*
STB-20	26	sts8	Hansen's test for parameter instability *K. Heinecke & C. Morris*
STB-21	33	sts9	Johansen's test for cointegration *K. Heinecke & C. Morris*

[zz] Not elsewhere classified

STB-19	30	zz3.3	Computerized index for the STB (Update) *W. Gould*
STB-20	32	zz3.4	Computerized index for the STB (Update) *W. Gould*
STB-21	40	zz3.5	Computerized index for the STB (Update) *W. Gould*
STB-22	31	zz3.6	Computerized index for the STB (Update) *W. Gould*
STB-23	29	zz3.7	Computerized index for the STB replaced in Stata 4.0 *W. Gould*
STB-19	31	zz4	Cumulative index for STB-13–STB-18

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	*qs*	questions and suggestions
dt	data sets	*tt*	teaching
gr	graphics	*zz*	not elsewhere classified
in	instruction		
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.

International Stata Distributors

International Stata users may also order subscriptions to the *Stata Technical Bulletin* from our International Stata Distributors.

Company: Dittrich & Partner Consulting
Address: Prinzenstrasse 2
D-42697 Solingen
Germany
Phone: +49 212-3390 99
Fax: +49 212-3390 90
Countries served: Austria, Germany

Company: Howching
Address: 11th Fl. 356 Fu-Shin N. Road
Taipei, Taiwan, R.O.C.
Phone: +886-2-505-0525
Fax: +886-2-503-1680
Countries served: Taiwan

Company: Metrika Consulting
Address: Ruddammsvagen 21
11421 Stockholm
Sweden
Phone: +46-708-163128
Fax: +46-8-6122383
Countries served: Baltic States, Denmark, Finland, Iceland, Norway, Sweden

Company: Oasis Systems BV
Address: Lekstraat 4
3433 ZB Nieuwegein
The Netherlands
Phone: +31 3402 66336
Fax: +31 3402 65844
Countries served: The Netherlands

Company: Ritme Informatique
Address: 34 boulevard Haussmann
75009 Paris, France
Phone: +33 1 42 46 00 42
Fax: +33 1 42 46 00 33
Countries served: Belgium, France, Luxembourg, Switzerland

Company: Timberlake Consultants
Address: 47 Hartfield Crescent
West Wickham
Kent BR4 9DW, U.K
Phone: +44 181 462 0495
Fax: +44 181 462 0493
Countries served: Eire, Portugal, U.K.