Tests for Departure from Normality

1 author:

Patrick Royston
University College London
401 PUBLICATIONS   **52,839** CITATIONS

Contents of this issue

an1.1	STB categories and insert codes (reprint)

Inserts in the STB are presently categorized as follows:

General Categories:

an	announcements	*ip*	instruction on programming
cc	communications & letters	*os*	operating system, hardware, &
dm	data management		interprogram communication
dt	data sets	*qs*	questions and suggestions
gr	graphics	*tt*	teaching
in	instruction	*zz*	not elsewhere classified

Statistical Categories:

sbe	biostatistics & epidemiology	*srd*	robust methods & statistical diagnostics
sed	exploratory data analysis	*ssa*	survival analysis
sg	general statistics	*ssi*	simulation & random numbers
smv	multivariate analysis	*sss*	social science & psychometrics
snp	nonparametric methods	*sts*	time-series, econometrics
sqc	quality control	*sxd*	experimental design
sqv	analysis of qualitative variables	*szz*	not elsewhere classified

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

crc1	CRC-provided support materials (reprint)

The materials that used to be described in the *Stata News* and provided on the Stata Support Disk are now moved to the STB. The materials provided in the \crc directory of the STB disk are the same, cumulative materials that were previously provided on the Stata Support Disk.

Inserts published in the *crc* category are included in the \crc directory and supported on our help line.

crc7	Corrections and changes in ado-files

logiodds: The reported pseudo-R^2 when logiodds is invoked with an if qualifier and missing data is incorrect. The problem is fixed on STB-2.

mdytoe, etc.: The date functions intentionally refused to create the requested result and instead produced an error message when the data did not appear to be in the specified format. For instance, 'datetof hdate, gen(hf)' would refuse to create hf if one of hdate's observations contained, say, "5/0/90" (an invalid *mm*/*dd*/*yy* date). The intention was to ensure that you knew there were problems that needed to be fixed. Other Stata commands, however, typically set the result to missing and continue. For instance, if myvar contained -1 and 4, 'gen newvar=sqrt(myvar)' would produce missing and 2 rather than refusing to take any of the square roots of myvar. On STB-2, the date functions have all been changed to adopt the standard behavior. The warning that there is a problem is via the missing values report. Typing 'datetof hdate, gen(hf)' would warn you "1 missing value generated" and define hf in the other observations.

survsum: The version distributed with Stata 2.1 sometimes calculates the median "incorrectly." Assume that the 50th percentile of the data occurs within a group of censored observations. The "correct" solution is to report the time associated with the earliest censoring, but survsum instead reports the last. (The median is normally defined as the time at which the survival curve first touches or crosses the 50% level.) This bug does not affect the reported mean, the survival curve calculated by survcurv, or kapmeier's Kaplan–Meier plot. A revised survsum procedure was silently included on STB-1 and a copy is also included on STB-2.

crc8	Actuarial or life-table analysis of time-to-event data

ltable is a rewrite of lftbl presented in *ssa1* by Henry Krakauer and John Stewart, Office of Research, Health Care Financing Administration. The advantages of ltable over lftbl are (1) it is supported by CRC, (2) it is more robust to user errors, and (3) it is command rather than menu driven. You may not consider (3) an advantage. Users wishing a menu-driven front-end should see *ssa1.1*. The syntax of ltable is

> ltable *timevar deadvar* [in *range*] [if *exp*] [, by(*groupvar*) intervals(#,#[,#,...])
>
> units({ days | weeks | months | years }) graph fsaving(*filename*[,replace])
>
> hsaving(*filename*[,replace]) fylabel[(...)] hylabel[(...)] *graph_options*]

ltable provides estimates of hazards and failures within interval, stratified by *groupvar* and including two tests of equality over strata if by() is specified. If graph is specified, ltable also produces a graph with confidence intervals for failures and hazards.

intervals(0,7,15,30,60,90,180,370,540,720) is assumed if intervals() is not specified. This is interpreted as the intervals $[0, 7)$, $[7, 15)$, $[15, 30)$, ..., $[720, \infty)$. If units() is specified, *timevar* is assumed to be coded in the specified units and temporarily recoded to days. Otherwise, *timevar* is used unmodified to determine the interval for each observation.

ltable performs the actuarial or life-table approach to the analysis of time-to-event data. This is particularly useful for analyzing large data sets. Times-to-events (failures and censorings or withdrawals without failure) are grouped into convenient intervals. The ratio of cases that failed to the number at risk is computed for each interval and, from that, the cumulative proportion of survivals and failures. One typically assumes that cases censored within the interval are at risk for half the interval, which ltable makes operational by assuming that half the censored cases are at risk in that interval.

The variables are coded in the usual way: *timevar* is a nonnegative variable indicating the time of failure or censoring for the observation. *deadvar* is a $\{0, 14\}$ dummy equal to 1 if the observation represents a real failure and 0 if it instead is censored. There is no assumption that *timevar* is coded in any particular units unless you specify the units() option—in which case you indicate the units of *timevar*. *timevar* is then temporarily recoded to days, which may make the default intervals() reasonable. Alternatively or in addition, you may specify the intervals explicitly. If you do not specify units(), specify intervals in the same units as *timevar*. If you do specify units(), specify intervals in days regardless of how *timevar* is coded.

If you specify the graph option, in addition to the failure and hazard tables, results are shown graphically. In addition to the standard graph-command options, such as symbol(), xlabel(), border, etc., you may specify fsaving(), the filename for saving the failure graph; fylabel(), the values to be labeled on the y-axis of the failure graph; hsaving(), the filename for saving the hazard graph; and hylabel(), the values to be labeled on the y-axis of the hazard graph.

For instance, to produce tables stratified by drug:

```
. ltable studytim died, by(drug)
```

| | | | | | | | Results of Life Table Analysis -- Failed |
Interval	Beg. Total	Deaths	Lost	Cum. Failed	Std. Error	Lower Limit	Upper Limit
drug 1							
0	20	8	8	0.0000	0.0000	0.0000	0.0000
7	12	7	8	0.4000	0.1826	0.7578	0.0422
15	4	4	4	0.7652	0.2464	1.2481	0.2823
drug 2							
0	14	1	2	0.0000	0.0000	0.0000	0.0000
7	12	2	5	0.0741	0.0770	0.2250	-0.0768
15	7	3	6	0.2504	0.1559	0.5559	-0.0550
30	1	0	1	0.6593	0.3722	1.3887	-0.0701
drug 3							
0	14	1	1	0.0000	0.0000	0.0000	0.0000
7	13	1	1	0.0714	0.0741	0.2167	-0.0738
15	12	3	7	0.1428	0.1013	0.3414	-0.0557
30	5	1	5	0.4000	0.2006	0.7931	0.0068

| | | | | | | | Results of Life Table Analysis -- Hazard |
Interval	Beg. Total	Cum. Failed	Std. Error	Hazard	Std. Error	Upper Limit	Lower Limit
drug 1							
0	20	0.0000	0.0000	0.0714	0.0244	0.1194	0.0235
7	12	0.4000	0.1826	0.1094	0.0372	0.1822	0.0365
15	4	0.7652	0.2464	.	.	.	.
drug 2							
0	14	0.0000	0.0000	0.0110	0.0110	0.0325	-0.0105
7	12	0.0741	0.0770	0.0263	0.0185	0.0626	-0.0100
15	7	0.2504	0.1559	0.0500	0.0268	0.1024	-0.0024
30	1	0.6593	0.3722	.	.	.	.
drug 3							
0	14	0.0000	0.0000	0.0106	0.0106	0.0313	-0.0101
7	13	0.0714	0.0741	0.0100	0.0100	0.0296	-0.0096
15	12	0.1428	0.1013	0.0235	0.0134	0.0497	-0.0027
30	5	0.4000	0.2006	.	.	.	.

```
Likelihood-ratio test statistic for homogeneity (group=drug):
Chi2( 2 ) = 20.02446,   P = .00004485

Logrank test of homogeneity (group=drug):
```

```
Group           Events          Predicted
---------------------------------------------
1               19              7.2459559
2                6              8.1984653
3                6              15.555579
Chi2( 2 ) =  25.526241 , P =  2.864e-06
```

To produce graphs as well as the tables in this last example, you would type 'ltable studytim died, by(drug) graph'.

crc9	Replacing coded missing values

One occasionally reads data sets where missing (e.g., failed to answer a survey question, or the data was not collected, or whatever) is coded with a special numerical value. Popular codings are 9, 99, $-]nn9$, -99, and the like. The numbers 9 and 99 have no special meaning to Stata, so the first thing you have to do with such data is to go through and replace the coded missing values with Stata's '.' missing value. You have to type 'replace x=. if x==99', say, replacing x with each and every variable in your data. mvdecode makes this process easier:

$$\text{mvdecode } varlist \left[\text{if } exp\right] \left[\text{in } range\right], \text{mv}(\#) \left[\text{quiet}\right]$$

mvdecode changes all occurrences of # to missing in the specified *varlist*. Thus, typing 'mvdecode _all, mv(99)' would translate 99 to Stata's missing value. Use this command cautiously since, even if 99 is not a special code, all 99's will be changed to missing.

Taking the flipside of the problem, one occasionally needs to export a data set to software that does not understand that '.' means missing, so one has to code missing with a special numerical value. The syntax of mvencode is

$$\text{mvencode } varlist \left[\text{if } exp\right] \left[\text{in } range\right], \text{mv}(\#) \left[\text{override quiet}\right]$$

mvencode changes all occurrences of missing to # in the specified *varlist*. Thus, typing 'mvencode _all, mv(99)' would change all missing values to 99.

mvencode is smart: It refuses to make the change if # (99 in this case) is already used in the data, so you can be certain that your coding is unique. mvencode will also automatically recast variables upward if necessary, so even if a variable is stored as a byte, its missing values can be recoded to, say, 999.

dm1	Date calculator

Manuelita Ureta, Texas A&M University

[The programs described below can be found in the \dm1 *directory of the* STB-2 *disk—Ed.]*

I often need a hand date calculator. That is, I need to know the date corresponding to, say, 47 days from some given date, or I need to know the number of days between two dates. dtadd and dtdiff solve this problem. dtadd calculates a future date:

$$\text{dtadd } curdate\ amt\ \left[\ \text{days} \mid \text{years}\ \right]$$

dtadd displays the future date (in *yymmdd* format) *amt* from *curdate*. *curdate* is also specified in *yymmdd* format, and *amt* is specified as either number of years or number of days:

```
. dtadd 910505 57 days
  Future date is:  910701
```

When specifying amt in years, the calculation is an approximation; dtadd assumes 365.25 days per year. For some applications, this may be incorrect.

dtdiff calculates the number of days between dates:

$$\text{dtdiff } date1\ date2$$

dtdiff displays the number of days between two formatted dates. Each date is specified in *yymmdd* format. The reported year estimate assumes 365.25 days per year:

```
. dtdiff 910505 910701

  Difference in Days is:       57
  Difference in Weeks is:    8.14
  Difference in Years is:   0.156
```

It should be noted that dtdiff calculates the number of days up to, but not including, the second date.

| gr2 | Importing Stata's Graphs Into MS-Word or Wordperfect |

Richard E. Deleon, San Francisco State Univ., and J. Theodore Anagnoson, Cal. State Univ., LA

[The Microsoft Word version mentioned in this insert refers to Word for DOS—the Windows version first necessitates the inclusion of an appropriate conversion driver—Ed.]

This article has two purposes—first to show you a small dataset that illustrates many points about the countries in the recent Gulf War, and second, to show you how to import a Stata graph directly into a Microsoft Word or into a Wordperfect document. First, we create a Stata file of data about the Gulf War countries. Here, I will simply describe and list the dataset, so that those of you who want to duplicate it can do so:

```
. describe

Contains data from gulfdat.dta
  Obs:     7 (max=  2315)            Gulf War Country Data
 Vars:     7 (max=    99)
   1. country      str15  %15s
   2. life         float  %9.0g    Life Expectancy at Birth, 1987
   3. gdpcap       float  %9.0g    Gross Dom Prod/Capita 1987
   4. literacy     float  %9.0g    Adult % literacy, 1985
   5. milgnp       float  %9.0g    Military Expend as % of GNP, 86
   6. ratmilhe     float  %9.0g    Ratio Mil. Health-Educ Spndg, 86
   7. armsimp      float  %9.0g    Arms Imports, 1987, US$ Million
Sorted by:

. list

          country  life  gdpcap  literacy  milgnp  ratmilhe  armsimp
   1.         Iraq    65    2400        89      32       711     5600
   2.       Jordan    67    3161        75    13.8       197      320
   3.         Iran    66    3300        51      20       377     1500
   4.  Saudi Arabia    64    8320        55    22.7       155     3800
   5.       Israel    76    9182        95    19.2       204     1600
   6.       Kuwait    73   13843        70     5.8        77      150
   7.         U.S.    76   17615        95     6.7      62.5     62.3
```

Now, we create two graphs and save them:

```
. graph life gdpcap, s([country]) xlab ylab ti("Life Expectancy by
  GDP Per Capita, 1987, Gulf Nations") saving(gr1)
. graph milgnp ratmilhe, s([country]) xlab ylab ti("Military $
  Compared to Health-Educ $, Gulf Nations") saving(gr2)
```

Then we exit from Stata. To insert the graphs into a Microsoft Word file, I first use `gphpen` to convert them to Lotus 123's *pic* format (see pp. 204–205 of the Stata Manual).

```
> gphpen gr1 /dpic.pen
> gphpen gr2 /dpic.pen
```

This command yields two files named `gr1.pic` and `gr2.pic`. These are then imported to the Microsoft Word file with the commands **ESC Library Link Graphics**, and then you specify the file name, etc. You can look at the page with **ESC Prnt preView**. *Figure 1* shows the U.S., Kuwait and Israel in the upper right as nations with relatively high life expectancies and high gross domestic products, with Jordan, Iran, and Iraq at the lower left with relatively low numbers.

Figure 2 shows the U.S. and Kuwait at the lower left as nations which spend a relatively low proportion of their GNPs on the military and at the same time have a low ratio of military spending to health and education spending. Iran, Israel, Saudi Arabia and Jordan are in the middle, but Iraq is in the upper right in a class by itself, with a high proportion of its GNP spent on the military and a very high ratio, compared with other countries, of military spending to health and education spending.

To import the same graphs into Wordperfect follow the outline below:

1. In DOS, execute 'gphpen *filename.gph* `/dpic.pen`'. That will create a new file, *filename*`.pic`.

2. In Wordperfect, locate where you want the *pic* version of the Stata graph to go within the document. Press **Graphics (Alt-F9)**. Then, in successive menus, . . .

3. Select (1) **Figure**.

4. Select (4) **Options** if you want to change default border style, placement of caption, etc. Then **F7** to return to document.

5. Select (1) **Create**.

6. Press (1) and enter `filename:\ `*filename*`.pic`

7. Select any other options you desire, including (8) **Edit**. Then **F7** to return to document.

8. Press **Shift-F7**, **6** to view document as it will appear when printed.

Life Expectancy by GDP/Per Capita, Gulf War Nations

Figure 1

Military $ Compared to Health/Educ $, Gulf Nations
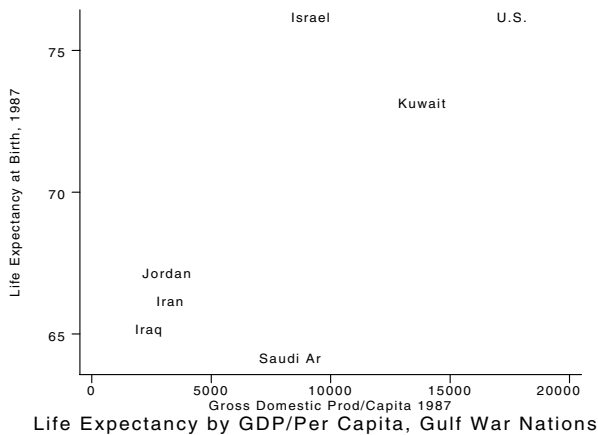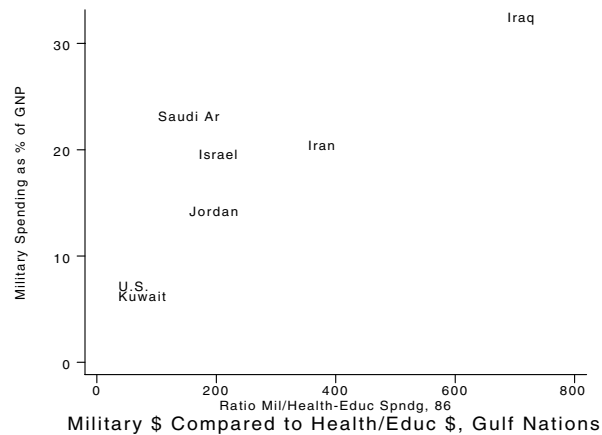
Figure 2

---

gr3	Crude 3-dimensional graphics

William Gould, CRC, FAX 213-393-7551

For those of you who have wished that Stata could draw 3-dimensional scatterplots, here is a solution. I have labeled the result "crude," although if you will flip to the end of this insert, you will see that the resulting graphs look reasonably good. They are "crude" in the sense that the commands to implement them have been written as ado-files rather than internal Stata commands and, to make them truly publication quality, you must use Stage to edit away a little of the chart junk the ado-file leaves behind. The basic command for drawing 3-d graphs is gr3:

gr3 [*xvar yvar zvar* [if *exp*] [in *range*] [, *options*]]

where the options are

option	Explanation	default
apen(#)	pen for axis	3
dpen(#)	pen for data	2
lpen(#)	pen for lines	1
tpen(#)	pen for text	3
spen(#)	pen for shadow	3
psize(#)	size of text	115
rot(#)	rotation angle	45 or as set by gr3set
elev(#)	elevation angle	45 or as set by gr3set
symbol(s)	see help graph	o
connect(l)	see help graph	.

in addition to all the other options allowed by graph.

gr3 draws 3-dimensional scatterplots. To graph the variables price, weight, and mpg, for instance, you type

```
. gr3 price weight mpg
```
(see Figure 1)

and the graph (Figure 1) will appear. The "crude" aspect of the graph is immediately apparent; the left and bottom edges have tick marks and labeling, chart junk that would usefully appear on a two-way scatter, that the ado-file could not suppress. In the remaining figures, I will use Stage to edit away the inappropriate labels.

Once a graph has been drawn, you can redisplay the graph by typing gr3 by itself. You can specify options either when you originally draw the graph or when you redraw it, so in constructing Figure 1, after obtaining a graph I liked, I typed 'gr3, saving(fig1)'. More usefully, you can change the rotation and elevation angles. The default is 45 degrees for each, but 30 and 60 degrees are also a good choice, as shown in Figure 2:

```
. gr3, rot(30) elev(60)
```
(see Figure 2)

gr3 achieves this magic by changing the data in memory. Once you are through looking at a 3-d graph, you type 'restore' to bring back your original data. Thus, you first draw a graph using gr3 followed by a *varlist*, you optionally redisplay the graph, varying the options using gr3 without a *varlist*, and finally, you restore your original data.

Two more commands help you tailor gr3 to your use: gr3set and gr3q. gr3q shows you the default and last-used settings:

```
. gr3q
-------------------------------------- Defaults
axis              tbl
shadow            off
rotation          45
elev              45
-------------------------------------- Last used
rotation          30
elev              60
--------------------------------------
```

gr3 provides four axis styles: tbl, a table-top that we have been using and that is my favorite; 0, a conventional x-y-z axis intersecting at $(0, 0, 0)$; min, a conventional x-y-z axis intersecting at $(x_{min}, y_{min}, z_{min})$; and out, an "outside" axis or a minimalist's table top. gr3 provides two ways of locating points in three-space: shadow off, which we have been using, draws vertical lines from the points to the x-y plane, whereas shadow on instead shows the shadows of the points on the x-y and y-z planes. shadow on works best with lots of data. Compare Figures 3 and 4:

```
. use citytemp, clear
. gr3set elev 55
. gr3 tempjan tempjuly heatdd              (see Figure 3)
. restore
. gr3set shadow on
. gr3 tempjan tempjuly heatdd              (see Figure 4)
```

Also note that in drawing these figures, I used gr3set to set the elevation to 55 degrees once and for all, saving myself the trouble of having to specify the elev(55) option on each of the gr3 commands. To set shadow (or the axis), you must use gr3set—there are no gr3 options to do this.

Finally, I will show you two more graphs which, while not pretty, can be useful:

```
. gr3set elev 45 rot 45 shadow off
. use auto, clear
. gr3 price weight mpg, s([model])        (see Figure 5)
. restore
. sort mpg weight price
. gr3 price weight mpg, c(l)               (see Figure 6)
```

Figure 5 is meant to demonstrate that graph's symbol() option works with gr3. However, if you are using a text symbol, you must specify the symbol() option at the time you originally draw the graph. If you subsequently redraw the graph, you can change the symbol, but only if you included, say, 's([model])' at the outset does gr3 know to keep the variable model for future use. Figure 6 is meant to demonstrate that graph's connect() option also works with gr3. The graph is not so silly. Notice that I first sorted the data on mpg, weight, and price. Comparing z (mpg) heights can be difficult when they are separated by considerable x-y distance. Looking at this graph, one can immediately see where the Seville fits with similar mpg data.

gr3 and its associated components are easy to use and completely documented by typing 'help gr3'. The materials are in the \gr3 directory of STB-2.
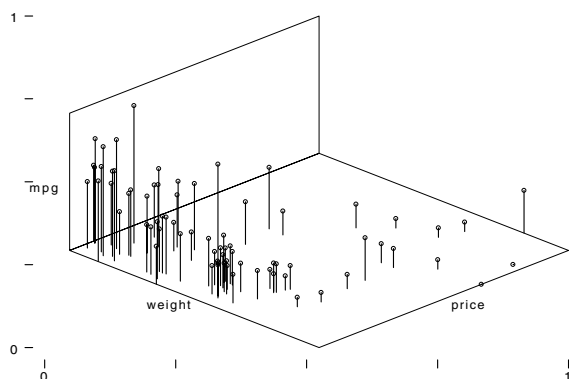
Figure 1
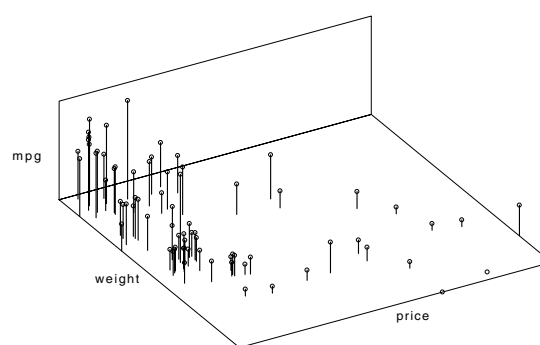
Figure 2
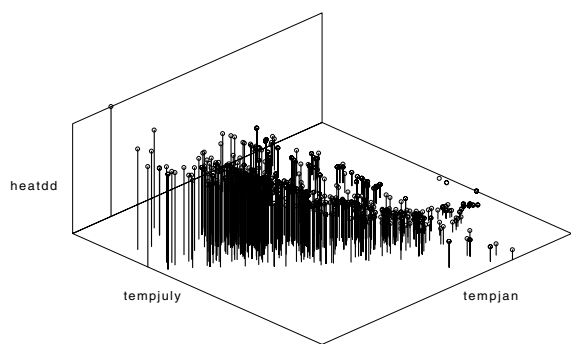
Figure 3
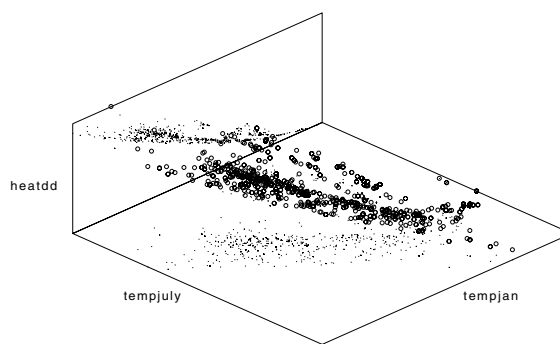
Figure 4
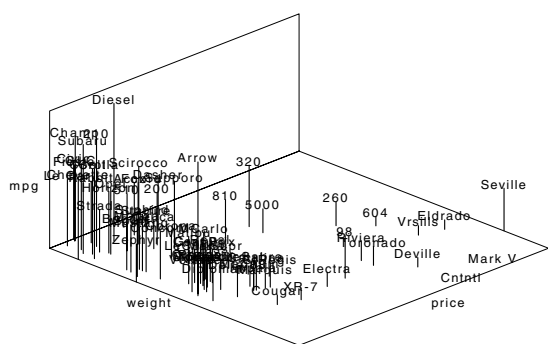
Figure 5
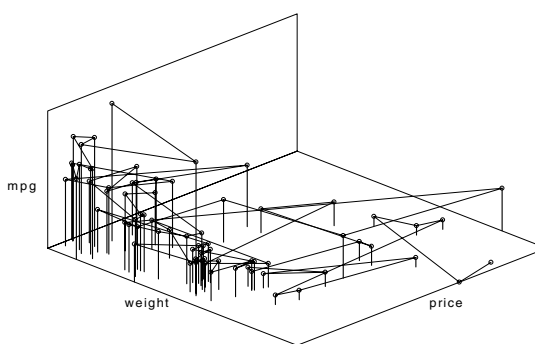
Figure 6

gr4	3-Dimensional contour plots using Stata & Stage

Gerard van de Kuilen, Oasis Decision Support Systems, The Netherlands FAX (011)-31 3402 65844

I have found it possible to write an ado-file using both Stata and Stage to produce 3-dimensional contour (wire-frame) plots. Figures 1 and 2 show the result of typing

```
. plot3d sin(x)^2-cos(y)^2
. plot3d -(exp(x)^(sin(x)^3) - cos(y)^3)
```

In the resulting graphs, the z-axis is vertical, the y-axis is horizontal, and the x-axis is "coming out of the paper."

The program is not as elegant as one might wish. For instance, whatever function is typed is graphed over the range $0 \leq x \leq 5$ and $0 \leq y \leq 5$ in 25 steps in each direction. The first few lines of plot3d.ado establish these limits, so you can change them. The beginning of the program reads:

```
program define plot3d
        mac def x_number=25
        mac def x_min=0
        mac def x_max=5
        mac def y_number=25
        mac def y_min=0
        mac def y_max=5
```

You can edit the program and change these constants. *[All materials are in the \gr4 directory of STB-2—Ed.]* The program creates files named pl1.gph, pl2.gph, pl3.gph, and pl4.gph in the current directory. The final result is always stored in pl4.gph.

z = sin(x)^2 - cos(y)^2
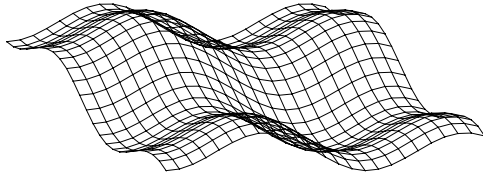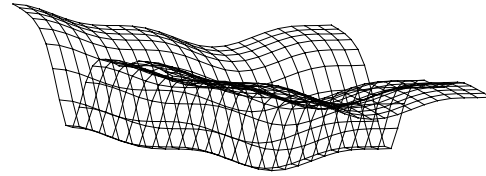
figure 1

z = -(exp(x)^(sin(x)^3) - cos(y)^3)

figure 2

gr5	Triangle graphic for soil texture

Francesco Danuso, Istituto di Produzione Vegetale, Udine, Italy. FAX (011)-39-432-558603

triatex.ado, supplied on the STB-2 disk together with two Stata data sets, is a graphical procedure which creates a soil texture triangle. The program should be adaptable for other purposes as well. The example used here relates to the relative mixture of clay and sand in twelve types of soil. The resultant graph plots the clay/sand values for each observation and their respective soil type percentage. The program does require a soil type data network called triatex.dta to format the experimental data. Help is available by typing 'help triatex' at the Stata prompt.

```
. use protex.dta

. describe

Contains data from protex.dta
  Obs:    10 (max= 77551)
 Vars:     2 (max=    99)
  1. clay          float  %9.0g
  2. sand          float  %9.0g
Sorted by:

. list

          clay        sand
  1.        20          30
  2.        18          45
  3.        10          80
  4.        35          15
  5.        50          20
  6.        25          33
  7.        40          20
  8.        14          30
  9.        10          55
 10.        30          50

. triatex

          TRIANGLE FOR SOIL TEXTURE
          (USDA classification scheme)

File name for the graphic (ENTER=do not save):
Symbol (O|S|T|o|d|p|.|i|[varname]):  [clay]
```

SOIL TEXTURE (USDA)
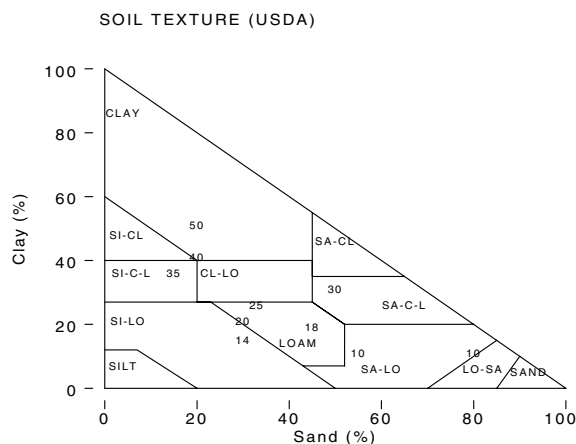
Figure 1

| os2 | Questions and answers about Stat/Transfer |

Steven Dubnoff, Circle Systems, FAX 206-328-4788

[Stat/Transfer is a program written by Circle Systems for converting data sets from one format to another. The program is available from CRC. Below, the most commonly asked questions are answered.—Ed.]

Q. I am using Borland's Quattro Pro Spreadsheet. Can I use Stat/Transfer to move my data into Stata?

A. Quattro Pro, Supercalc, Excel, and virtually every other spreadsheet program can read and write spreadsheets in Lotus "WK1" format. For example, in Quattro, you need only put a "WK1" extension on your output file name in order to write a WK1 file instead of Quattro's native format. Then, using Stat/Transfer, you can easily transfer the data into Stata.

Q. How can I control the size and data type of my Stata variables?

A. Stat/Transfer will, of course, convert numeric variables into Stata numbers and character variables into strings. When translating numeric variables, Stat/Transfer looks at both the print width and number of decimal places in order to determine what kind of Stata variable to create for each of your input variables. By controlling these in your input data set, you can control how Stat/Transfer converts each of your variables.

Based on the information it has, Stat/Transfer will attempt to minimize the size of your Stata file, according to the following rules: All variables which are formatted with one or more decimal places are translated into Stata floats. Input variables with zero decimal places are translated into various flavors of integers. Input variables with widths of one or two bytes are translated into Stata byte variables; variables with widths of three or four bytes are translated into two-byte integers (Stata ints); and variables with widths greater than four bytes are translated into four-byte integers (Stata longs).

You can use these rules to control how Stat/Transfer will treat each of your variables. If, for instance, you are transferring from a worksheet and you want to minimize the size of your input data set, make sure that each column is only as wide as it needs to be to display your variable and that, for integer variables, the first row is formatted with zero decimal places.

Q. How do I move large mainframe data sets into Stata?

A. The SPSS export format is usually the most convenient, as it preserves missing data, variables labels, and value labels. All of this information will be transferred automatically to your Stata data set. If your data are in SAS format, SPSS distributes a program, TOSPSS which will perform the translation from SAS to SPSS Export format.

Once your data are in SPSS format, you can transfer them via modem to your PC using the Kermit file transfer program which is included on your Stat/Transfer disk.

Because Stat/Transfer can handle data sets with as many as 1,000 variables, you can use Export files as a convenient storage medium for large data sets. You can then use Stat/Transfer to bring into Stata just the variables you need.

qs2	Why is the cubic spline so-called?

Ted Anagnoson, California State University, Los Angeles

Q. Stata has a curve smoothing function called "cubic spline." Why is it so-called?

A. The cubic spline is an instance of the more general method of "piece-wise" curve fitting. Its coefficients are adjusted in such a manner that different forms of the function fit various subdivisions of the exact data. Each subdivision is a different polynomial. In earlier times, draftsmen used a steel strip, positioned by weights, to join points on the "multi"-polynomial curve in an attempt to smooth it. This device was termed a "spline." However, it was found that the shape of the curve between each weight was a cubic polynomial due to the elastic deflection of the steel. Hence the reference to the procedure as "cubic spline."—*Ed.*

qs3	Request for confidence intervals for proportions

Ivan Zavala, Ph.D., Mexico City, Mexico

Q. Do you have or can you do a Stata ado-file to calculate confidence intervals for proportions?

A. I spoke to Bill Rogers at CRC and he agreed to write such an ado-file for a future STB.—*Ed.*

sbe2	Bailey–Makeham survival model

William Rogers, CRC, FAX 213-393-7551

(Under contract 500-90-0048 from the Health Care Financing Agency, a program has been implemented in ANSI C to estimate the Bailey–Makeham survival model, which will run within Stata or stand-alone. Also implemented is a set of Stata ado-files that make the Bailey–Makeham extensions appear to be an internal part of Stata. The source code and ado-files have been placed in the public domain. Included in the `\sbe2` directory is (1) full documentation, of which the below is an extract; (2) an executable (`.exe`) for 80286, 80386, or 80486 computers; (3) the source code; and (4) the ado files. Unix users can ignore the `.exe` file and recompile the source code. For DOS users, everything is ready to run.

The full documentation contains three sections: (1) The Bailey–Makeham model and the numerical strategy for estimation; (2) ASCII user's guide for the model; and (3) Stata user's guide for the model. Below is reprinted section 3 of the Report.)

The syntax of the `bailey` command is

$$\texttt{bailey } outcome\ varlist\ \big[\texttt{if } exp\big]\ \big[\texttt{in } range\big]\ \big[\texttt{=}exp\big]\ \big[\texttt{, dead}(varname)$$
$$\texttt{interval}(\#\ or\ varname)\ \texttt{trace autofix}(\#)\ \texttt{nocons rescale rhomax}(\#)\ \texttt{lambda}(\#)$$
$$\texttt{fxalpha}(varlist)\ \texttt{fxgamma}(varlist)\ \texttt{fxdelta}(varlist)\ \texttt{svalpha}(svlist)\ \texttt{svgamma}(svlist)\ \texttt{svdelta}(svlist)\ \big]$$

See Bailey (1988) for a complete description of this model. The model is a generalization of the exponential distribution and has hazard function

$$\alpha e^{-\gamma t} + \delta$$

where α, γ, and δ are all functions of X, the vector of covariates described in the *varlist*. The quantities α, γ, and δ are termed "structural parameters" since all of the effect of the covariates is represented by them. The functional form of these structural parameters is

$$\alpha = e^{\Sigma \alpha_i x_i}$$
$$\gamma = e^{\Sigma \gamma_i x_i}$$
$$\delta = e^{\Sigma \delta_i x_i}$$

The data may be actual failures or censored observations. The time from the start of observation to failure or censoring is given by the variable *outcome*. If there are any censored observations, you must specify the option `dead()` with a variable that is 1 if the case is a death and 0 if it is censored. If the observation is a death, it is assumed to fail in the interval starting at time $T : (T, T + \texttt{interval}())$.

The value of `interval()` may be a number or a variable name. It represents the graininess of the observation in the units of the outcome. The default is 1 unit. For example, if the survival times are measured in days, the graininess is assumed to be 1 day. In other words, you know the survival time to one day, but not closer than that. Although there is little difference between a fine-grained measurement and a continuous measurement, the likelihood calculated using the fine-grained approximation is simpler. If the option is specified as a number, the program assumes that this number applies to each case. If it is a variable, then the value of the variable applies to each case. One important special case is `interval(0.00274)`, which is $1/365$ of a year. In other words, this would be saying that the data are specified in years, but derived from data measured in days. Note that the value of `interval()` and the units of measurement affect the likelihood value, but do not change the nature of the solution.

Each of the variables in *varlist* applies to each structural parameter unless otherwise specified. This process is called "fixing." Variables are fixed for a structural parameter by including them on its fixed list. The list of variables specified in `fxalpha()` is fixed for α, the list in `fxgamma()` is fixed for γ, and the list in `fxdelta()` is fixed for δ. If a variable is not explicitly fixed, it is assumed to be free, or available to be maximized.

Starting values are defined by specifying the options `svalpha(`*svlist*`)`, `svgamma(`*svlist*`)`, or `svdelta(`*svlist*`)`. The argument *svlist* is a series of statements of the form *variable=#*. If starting values are not specified, they start at other values, usually 0.

Other options are as follows:

`trace`: Includes a trace of the parameter values at each maximization step.

`nocons`: Excludes the automatic constant term from the equation. Useful if there is no constant or if a mutually exclusive set of dummy variables is employed.

`rescale`: Rescales the likelihood by the sum of the weights (specified by the =*exp* option). This option is not advised, but is included for compatibility with other vendors' software. If this option is not used, the weights are treated like frequency weights. That is, a weight of 2 means there were two observations like this that were represented by a single line of the data. A third meaning of weights is that they represent inverse sampling probabilities. If this interpretation is used, the standard errors produced by this program will not be correct. However, it is better to rescale than not.

`maxiter`: Maximum number of iterations allowed.

`autofix(#)`: If this option is specified, then the program will automatically fix (stop) a coefficient that is rapidly changing, but not having an appreciable effect on the likelihood function. The purpose of this option is to gracefully deal with collinear and infinite coefficients. This option will kick in only if the change in the likelihood function is smaller than the given value. If this option is not specified, the program will make recommendations but will not take any action.

`rhomax(#)`: This parameter describes the extent to which we are willing to extrapolate in the modified Marquardt maximization. In the Marquardt method, a Newton step (or a ridge-like approximation thereto) is attempted first. The actual (log) likelihood gain is computed. This actual gain is combined with the gradient to calculate an optimal step via a quadratic approximation. This optimal stepsize is constrained to be at most `rhomax` times as large as the original Newton step. The default is 5; `rhomax` should always be greater than 1. A larger value is more aggressive.

`lambda(#)`: This parameter describes the (starting) height of the ridge that is used if needed in the modified Marquardt method whenever the 2nd derivative is non-positive definite. The value is modified during the iterations in light of experience. The default is 1.0.

Example

In this dataset of 200 hospitalized cancer patients, we measure the survival time as a function of the Karnofsky Performance Scale:

```
. summ survive karnofsk
Variable |   Obs        Mean    Std. Dev.       Min        Max
---------+-----------------------------------------------------
 survive |   200       46.15     61.3805          1        345
karnofsk |   200        49.6    27.79755         10        100
. cox survive karnofsk, dead(dead)
Iteration 0:  Log Likelihood =-865.73173
Iteration 1:  Log Likelihood =-852.31805
Iteration 2:  Log Likelihood =-852.31781
Cox regression                              Number of obs =     200
                                            chi2(1)       =   26.83
Log Likelihood =-852.31781                  Prob > chi2   = 0.0000
Variable |  Coefficient   Std. Error      t    Prob > |t|     Mean
---------+--------------------------------------------------------
  survive|                                                   46.15
     dead|                                                       1
---------+--------------------------------------------------------
  karnofsk|   -.015111     .0029536    -5.116    0.000       49.6
---------+--------------------------------------------------------
. bailey survive karnofsk, dead(dead) autofix(.01)
Iteration   0: Log Likelihood =          -959.1589
Iteration   1: Log Likelihood =          -955.2354
Iteration   2: Log Likelihood =          -954.4966
Iteration   3: Log Likelihood =          -954.4889
Iteration   4: Log Likelihood =          -954.4889
```

```
Iteration   0: Log Likelihood =       -954.4889
Iteration   1: Log Likelihood =       -946.3064
Iteration   2: Log Likelihood =       -941.3947
Iteration   3: Log Likelihood =       -937.1823
Iteration   4: Log Likelihood =       -936.9542
Iteration   5: Log Likelihood =       -936.9427
Iteration   6: Log Likelihood =       -936.9426
Iteration   7: Log Likelihood =       -936.9426

(convergence achieved)
Bailey-Makeham Survival Model 1.0
Log Likelihood (C) =      -954.489   Number of observations =          200
Chi2(  3)       =           35.093   Log Likelihood          =      -936.943
Structural      |
parameter   Var. |      Coef.     Std. Err.      t      Sig.         Mean
-----------------+-------------------------------------------------------------
   alpha karnofsk |  -0.0287193  0.00898029   -3.198   0.0016         49.6
   gamma karnofsk |   0.0338422  0.0177919     1.902   0.0586         49.6
   delta karnofsk |   0.00799231 0.0107872     0.741   0.4596         49.6
   alpha    _cons |   -2.47772   0.25908       -9.564   0.0000            1
   gamma    _cons |   -5.52475   1.19964       -4.605   0.0000            1
   delta    _cons |   -4.98935   0.893983      -5.581   0.0000            1
-----------------+-------------------------------------------------------------
```

(Note, the iterations restart because the optimization procedure first fits a reference model consisting of a constant term for each of the structural parameters. Only after that model has converged are the other parameters freed for optimization.)

In the coefficient report, note that the log likelihood of the reference model and the chi-square of the final model with respect to the reference model are printed on the left-hand side of the header.

We conclude from this analysis that patients with a high Karnofsky rating are less likely to die quickly, and that their hazard decays to its long term value sooner. However, patients with a high Karnofsky rating do not have any better long-term survival rates.

Predictions

There are two types of predictions available for individuals in the dataset: failure probabilities at specific points in time and predictions of the structural parameters and their standard errors. The pertinent variable names to receive these quantities are specified as part of the options to the mpredict command, unlike most other Stata commands. The syntax of mpredict is

$$\text{mpredict, pparm}(parmlist) \text{ ptime}(faillist)$$

where *parmlist* has one or more elements of the form

$$varname[\texttt{alpha}] \mid varname[\texttt{gamma}] \mid varname[\texttt{delta}] \mid$$

$$varname[\texttt{var[alpha]}] \mid varname[\texttt{var[gamma]}] \mid varname[\texttt{var[delta]}] \mid$$

$$varname[\texttt{cov[alpha,gamma]}] \mid varname[\texttt{cov[alpha,delta]}] \mid varname[\texttt{cov[gamma,delta]}]$$

and *faillist* has one or more elements of the form *varname*[time]. Note that what is typed in the square brackets indicates what is being predicted; what is in front of the square brackets is the name of a new variable to be created containing that prediction. Multiple elements are separated by spaces. In the above Karnofsky example, we might give the command

```
. mpredict, pparm(alpha1[alpha] gamma1[gamma] delta1[delta]) ptime(p6[180])
```

to create four new variables, alpha1, gamma1, delta1, and p6. These are the estimates of the structural parameters for each observation and the estimated 180-day failure probability.

Surface Analysis

A surface analysis is performed by typing the msurface command, which takes no arguments. A surface analysis is an examination of the likelihood surface if each parameter is moved 1 and 2 standard errors above or below the estimated parameter, holding all other parameters fixed. The change in log likelihood is printed, along with a number in parentheses which is the ratio of the actual change to that predicted by the quadratic approximation, using the second derivative matrix.

If the numbers in parentheses are larger than 2 or below 0.5, the quadratic approximation at the maximum is definitely poor. This points to problems with the estimated standard errors and possible problems with multiple or infinite solutions.

References

Bailey, R. C. 1977, Moments for a modified Makeham Law of mortality. Naval Medical Research Institute Technical Report, Bethesda MD.

——. 1988. The Makeham Model for Analysis of Survival Data.

Bailey, R. C., L. D. Homer, J. P. Summe. 1977. A proposal for the analysis of kidney graft survival. *Transplantation* 24: 309–315

Bard. 1974. *Nonlinear Parameter Estimation*. Academic Press.

Krakauer, H. K., R. C. Bailey. 1991. Epidemiologic oversight of the medical care provided to Medicare beneficiaries. *Statistics in Medicine* 10.

Press, W. H., S. A. Teukolsky, B. P. Flannery, W. T. Vetterling. 1990. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge: Cambridge University Press.

Rogers, W. H., D. Draper, K. L. Kahn, E. B. Keeler, L. V. Rubenstein, J. Kosekoff, and R. H. Brook. 1990. Quality of care before and after implementation of the DRG-based prospective payment system. *Journal of the American Medical Association* 264(15): 1989–1994.

Stewart, G. W. 1973. *Introduction to Matrix Computation*. Academic Press.

Summe, J. P. 198?. Bailey Makeham Instructions for SAS mainframe. Private Communication.

---

sed2	Ladder-of-Powers variable transformation

William Gould (CRC) and Joseph Hilbe (STB)

[Many statistical procedures assume, among other things, that the variables being analyzed come from a normal distribution. If a variable is highly skewed it must be normalized; i.e., we mathematically reshape the variable so that it better approximates a normal or Gaussian distribution. sed2 and sed3 are programs which attempt to aid the analyst in selecting the proper transformation. Both yield criteria information and provide graphical support. The foremost difference between the two is that sed2 evaluates transformation normality by means of `sktest` *whereas sed3 remains strictly empirical.—Ed.]*

The commands `ladder` and `gladder` help find a transform from the ladder of powers (see Tukey 1977 or Hamilton 1990) to "normalize" (make Gaussian) a variable. `ladder` uses an arithmetic whereas `gladder` uses a graphical approach. `ladder` tests each transformation from the ladder of powers by means of `sktest` (also see *sg3* and *sg3.1*). Its syntax is

$$\text{ladder } \textit{varname} \ \big[\text{if } \textit{exp}\big] \ \big[\text{in } \textit{range}\big] \ \big[\text{, generate}(\textit{newvar})\big]$$

The `generate()` option is probably a bad idea as it is easily misused. If specified, *newvar* will be created that corresponds to the minimum chi-square value, where minimum is quite literally interpreted. `ladder` will gladly ignore nearly equal values of chi-square that allow, perhaps, simpler interpretations. For instance:

```
. ladder mpg, gen(mpgx)
    Transformation        formula       Chi-sq(2)    P(Chi-sq)
    ------------------------------------------------------------
    cube                  mpg^3           238.53       0.000
    square                mpg^2            80.88       0.000
    raw                   mpg              18.38       0.000
    square-root           sqrt(mpg)         6.70       0.035
    log                   log(mpg)          1.57       0.456
    reciprocal root       1/sqrt(mpg)       0.27       0.874
    reciprocal            1/mpg             1.83       0.400
    reciprocal square     1/(mpg^2)        17.11       0.000
    reciprocal cube       1/(mpg^3)        65.20       0.000

       <<mpgx = 1/sqrt(mpg)   generated>>
```

Note that `ladder` locked on to `1/sqrt(mpg)` when either `log(mpg)` or `1/mpg` would be equally good. In this case, `1/mpg` would probably be preferred since it has a natural interpretation, namely `gpm` (gallons per mile). You do not have to use the `generate()` option. Omitting it would still produce the table. Notice that, in making the automatic calculation, the sense of the variable may be reversed. Most ladder-of-power diagrams place negation operators in front of reciprocal transformations (e.g., $-1/\text{mpg}$) to preserve sense. This, however, often makes for inconvenient interpretation.

`gladder` works almost identically to `ladder`, but rather than producing a table, it produces an array of nine histograms, one for each of the ladder-of-power transforms. The syntax is

$$\text{gladder } \textit{varname} \ \big[\text{if } \textit{exp}\big] \ \big[\text{in } \textit{range}\big] \ \big[\text{, bin}(\#) \ \textit{graph\_options}\big]$$

If `bin()` is not specified, the default is set according to the suggestion made in *gr1*. The calculation is made in terms of the number of unique values of *varname*, and $\sqrt{n}$ is used in preference to $\log_{10} n$ if the former is smaller. Thus, typing '`gladder mpg`' results in

Histograms by Transformation
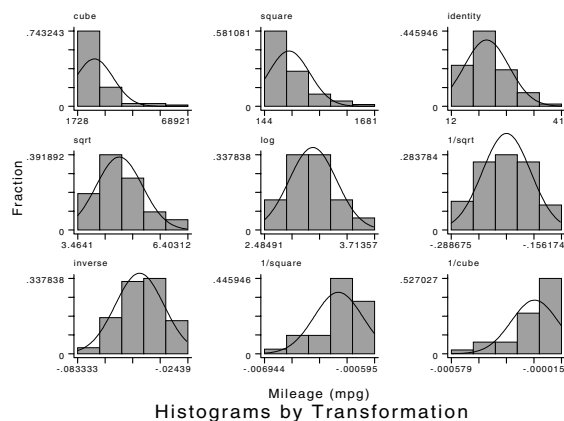
References

Tukey, J. W. 1977. *Exploratory Data Analysis.* Reading, MA: Addison–Wesley.

Hamilton, L. C. 1990. *Statistics with Stata.* Pacific Grove, CA: Brooks/Cole.

sed3	Variable transformation and evaluation

Thomas Findley, M.D., Ph.D., Kessler Institute for Rehabilitation, West Orange, New Jersey

`transfor.do` provides a host of univariate statistics regarding seven possible transformations from the ladder of powers. Transformation information is displayed for the cube, square, identity (raw), square root, natural log, negative reciprocal root, and negative reciprocal. For each transformation, reported are the mean, median skewness, kurtosis, standard deviation, interquartile range/1.35, 90% range calculated as mean $\pm\, 1.6$S.D., and the actual 90% range (5% to 95%). Also for each, a histogram with an overlaying normal distribution curve and a one-way graph with overlaid box plots are shown. Graphs are automatically shown and saved on disk. The syntax is

$$\texttt{run transfor } \textit{varname } \big[\textit{varname2}\big]$$

where *varname* is the variable to be transformed and *varname2* is an optional variable you may want *varname* graphed against. The tabular output is too long to present here, but the final two graphs when run on the `mpg` variable in the `auto.dta` are shown below.

Figure 1

Figure 2

sg3.1	Tests for departure from normality

Patrick Royston, Royal Postgraduate Medical School, London, FAX (011)-44-81-740 3119

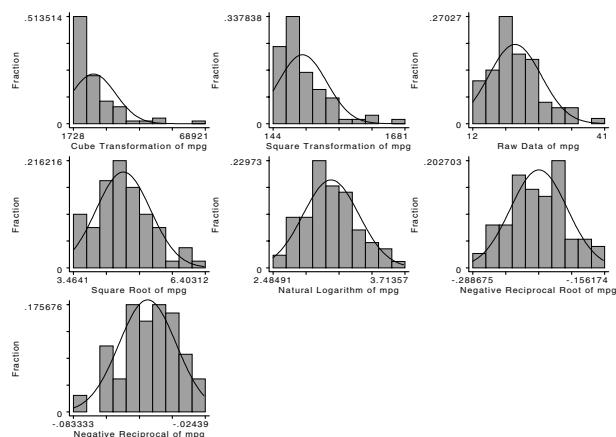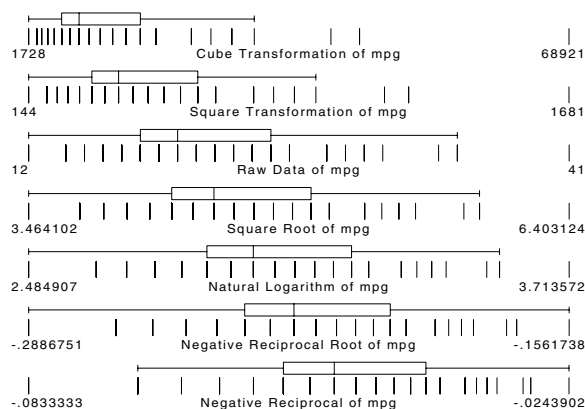*[Dr. Royston has evidently been interested in the development and algorithmic interpretation of tests and estimates of departure from normality for many years. He included with his submission a list of 16 published papers on the subject. Readers interested in this subject should see the file* \sg3.1\readme *for the list.—Ed.]*

Like Gould, I had never heard of the Bera–Jarque test until recently. However, since I believe sktest to be poor, for reasons I will supply below, the B–J test must also be defective since it is essentially the same as sktest.

Before turning to my justification, I will note what I assume are typographical errors in *sg3*: In paragraph 2 the expression $Z_3 = s_3\sqrt{\bar{x}}/\sqrt{6}$ and $Z_4 = (s_4 - 3)\sqrt{\bar{x}}/\sqrt{24}$ should read $Z_3 = s_3\sqrt{n}/\sqrt{6}$ and $Z_4 = (s_4 - 3)\sqrt{n}/\sqrt{24}$, n being the sample size. *[This is true—Ed.]*

I agree that if Z_1 and Z_2 are *independently* distributed as $N(0,1)$, then the combined test statistic $Z_1^2 + Z_2^2$ is distributed as χ^2 with 2 degrees of freedom. However, although s_3 and s_4 for samples from normal populations are each distributed as $N(0,1)$ in large samples (i.e., asymptotically), unfortunately they are not normal in samples of the size one encounters in applied work ($n < 1000$, say, depending on your field). Interestingly, s_3 and s_4, though uncorrelated, are not independent (Pearson, D'Agostino & Bowman 1977, p. 234), an observation D'Agostino seems to overlook in the 1990 *American Statistician* article despite having recognised it in 1977.

The results presented by Gould do not address this problem. I will argue that the power comparisons of the sort in Gould's table are not satisfactory unless the rejection probabilities under the null hypothesis (normality) are close to their nominal values. The rejection probabilities of the sktest, B-J, and D'A tests for samples from normal populations of size 100 are not terribly close to the nominal 0.1, 0.05, and 0.01 levels, so one cannot validly compare their power with that of other tests. (Admittedly, Gould shied away from any such interpretation, but other STB readers might not.)

Even though among themselves, sktest, B–J, and D'A seem to have similar rejection probabilities at $n = 100$, the effects of the differences on power are hard to assess and may be greater than one thinks. For a given test, the higher the rejection probability, the higher the apparent power.

Ideally, the null distribution of a test statistic should be known or should be reasonably accurately approximated, either analytically, or empirically using Monte Carlo simulation. Since the null distribution is often intractable, one usually has to use the simulation approach. Figures 1a and 1b show the empirical distribution of four test statistics for $n = 7$ and $n = 100$. D'A is D'Agostino et al. (1990) as quoted in *sg3* (not defined for $n < 20$ so it only appears for $n = 100$), SKTEST is sktest, W' is the Shapiro–Francia statistic (the squared correlation between the ordered data and the expected normal order statistics), and W is the Shapiro–Wilk test. Both W' and W are well-recognised as powerful "omnibus" (general purpose) tests. Each statistic has been "normalized" to have null distribution $N(0,1)$. For D'A and sktest, I have converted the (assumed) $\chi^2(2)$ statistic to $N(0,1)$ via the chi-square distribution function and the inverse normal distribution function (invnorm() in Stata-speak). For W' and W, I have used my published normalization methods (Royston 1983, 1982). In all cases, large positive values of the test statistic are regarded as evidence of "significant" departures from normality. Large negative values have no interest (unless cheating is suspected!).

All results in Figures 1a and 1b were obtained from simulation runs of length 10,000 as in *sg3*. (I used a 486 PC and my own venerable and much modified FORTRAN program which, being purpose-built and of course compiled, is many times faster than Stata.) The vertical axis y represents the difference between the empirical quantiles of each test statistic (which, apart from sampling variation, should agree with $N(0,1)$) and the corresponding $N(0,1)$ quantiles. Ideally all the tests should yield the line $y = 0$. Positive values of y imply the quantiles are higher than those of $N(0,1)$ and will produce inflated rejection probabilities if the normalized statistic is taken at face value.

As can be seen, sktest is grossly different from $N(0,1)$ for both sample sizes (the actual rejection probabilities are discussed below). Interestingly (and, though familiar with the D'Agostino test, called K^2 in the 1977 article mentioned above, I did not realize this), the D'A test is also inaccurate in the critical upper tail, rejecting too often, as the table in *sg3* suggests. Presumably this is due to the non-independence of s_3 and s_4. Both W' and W are reasonably accurate, especially the latter.

Figures 2a and 2b show the actual rejection probabilities at nominal 0.05 and 0.01 significance levels for the four tests over the range $7 \le n \le 100$ ($20 \le n \le 100$ for D'A). sktest's rejection rate happens to be close to the nominal 0.05 for $n = 100$ and to the nominal 0.01 at about $n = 15$, but otherwise is erratic. D'A always over-rejects. W' and W are close to nominal, W' rejecting slightly too often at the 0.05 level for $n > 20$.

My conclusion from Figures 1 and 2 is that neither sktest (and by implication Bera–Jarque) nor the D'A test are satisfactory as they stand. I therefore feel that Stata programs which quote P values for any of them are misleading, particularly for sktest, which should be withdrawn, even if the problems mentioned by Gould with sktest detecting departures from the uniform distribution did not exist.

I will supply as ado-files the alternative Shapiro–Wilk W and Shapiro–Francia W' tests for the next issue of the STB.

References

D'Agostino, R. B., A. Balanger, and R. B. D'Agostino, Jr. 1990. A suggestion for using powerful and informative tests of normality. *The American Statistician* 44(4): 316–321.

Gould, W. 1991. sg3: Skewness and kurtosis tests of normality. *Stata Technical Bulletin* 1: 20–21.

Pearson, E. S., R. B. D'Agostino, and K. O. Bowman. 1977. Tests of departure from normality: comparison of powers. *Biometrika* 64: 231–246.

Royston, J. P. 1982. An extension of Shapiro and Wilk's W test for normality to large samples. *Applied Statistics* 31: 115–124.

Royston, J. P. 1983. A simple method for evaluating the Shapiro–Francia W' test of nonnormality. *Statistician* 32: 297–300.
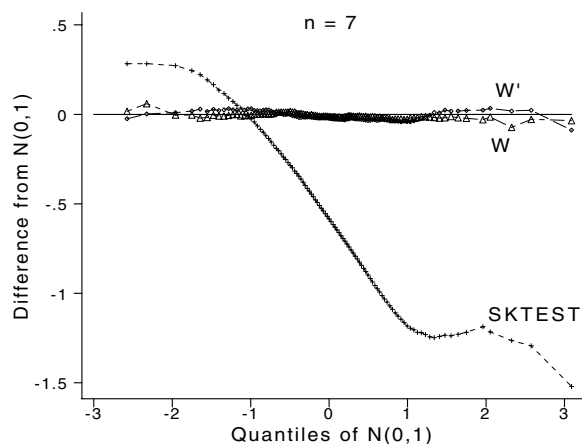
Figure 1a
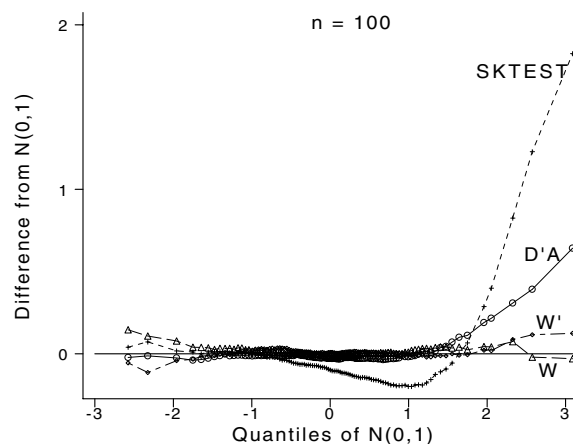
Figure 1b
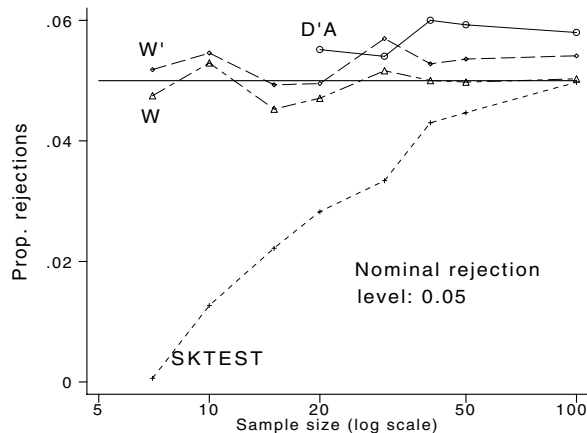
Figure 2a
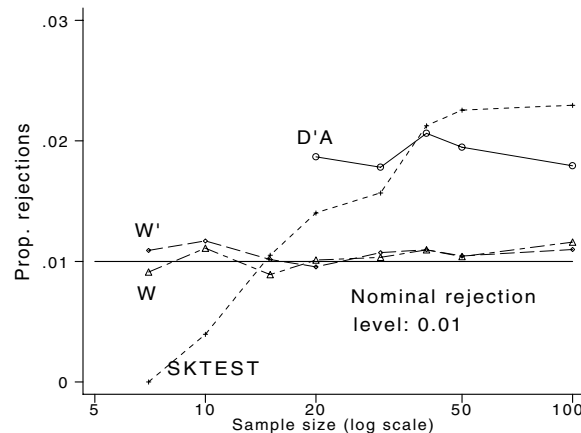
Figure 2b

snp1	Kolmogorov–Smirnov one and two variable tests

Lawrence L. Giventer, California State University, Stanislaus, FAX 209-667-3333

The Kolmogorov–Smirnov One Variable Test. The Kolmogorov–Smirnov one variable test compares the observed frequency distribution of a variable measured on an ordinal scale to an expected theoretical frequency distribution. Here is an example of how it works in Stata. A study of grand jurors in Alameda County, California compared the age distribution of jurors with the age distribution of the general population to see if the jury panels were representative. The variable "age" is grouped into four ordinal categories. The county-wide age distribution is known from census data.

```
        DISTRIBUTION OF GRAND JURORS AND ELIGIBLE
           COUNTY-WIDE POPULATION BY AGE GROUP,
           ALAMEDA COUNTY, CALIFORNIA, 1959-1969
    +-----------------+-------------+------------+
    |                 | Number of   | County-wide |
    | Row      Age    | grand jurors| proportion |
    | no.      group  |      1      |     2      |
    +-----------------+-------------+------------+
    | 1     21 - 40   |      5      |    0.42    |
    | 2     41 - 50   |      9      |    0.23    |
    | 3     51 - 60   |     19      |    0.16    |
    | 4     61 and up |     33      |    0.19    |
    +-----------------+-------------+------------+
    | Total           |     66      |    1.00    |
    +-----------------+-------------+------------+
```

Source: William G. Prahl, ``The Civil Petitioner's
Right to Representative Grand Juries and a Statistical
Method of Showing Discrimination in Jury Selection Cases
Generally'', UCLA Law Review, vol. 20 (1973), p. 615.

The Kolmogorov–Smirnov D statistic is found by comparing the observed cumulative relative frequencies against the cumulative relative frequencies of the expected distribution.

```
      DISTRIBUTION OF GRAND JURORS BY AGE GROUP (ALAMEDA COUNTY, CA, 1959-1969):
                      COMPUTATIONS FOR THE KOLMOGOROV-SMIRNOV TEST
  +---------+-----------------------+----------------------+--------------------+
  |         |                       |                      | Difference between  |
  |         |            Observed    |            Expected  | observed and        |
  |         | Observed  cumulative  | Expected  cumulative | expected            |
  |  Age    | absolute  relative    | relative  relative   | cumulative relative |
  | group   | frequency frequency   | frequency frequency  | frequencies         |
  +---------+-----------------------+----------------------+--------------------+
  |21 - 40  |    5       0.08       |  0.42      0.42      |      -0.34          |
  |41 - 50  |    9       0.21       |  0.23      0.65      |      -0.44          |
  |51 - 60  |   19       0.50       |  0.16      0.81      |      -0.31          |
  |61 and up|   33       1.00       |  0.19      1.00      |       0.00          |
  +---------+-----------------------+----------------------+--------------------+
```

The maximum difference between the observed and expected cumulative relative frequency distributions, regardless of whether it is positive or negative, is the calculated value of the Kolmogorov–Smirnov D statistic. In this example, D equals 0.44. This value can be compared against a published table of critical values to determine statistical significance. We conclude that in terms of age the grand jurors were not representative of the eligible population in Alameda County.

The procedure in Stata is not quite as straightforward. First, we need to input the table of observed frequencies and expected relative proportions. This is done by specifying the column number, row number, and cell contents, respectively. The column number distinguishes the observed and expected distributions and the row number identifies the age group.

```
. input col agegroup freq
           col   agegroup       freq
  1.         1          1          5
  2.         1          2          9
  3.         1          3         19
  4.         1          4         33
  5.         2          1        .42
  6.         2          2        .23
  7.         2          3        .16
  8.         2          4        .19
```

Next, we need to express the expected absolute frequency by multiplying the expected relative frequency for each ordinal category by the total number of observations.

```
. replace freq=.42*66 in 5
(1 change made)
. replace freq=.23*66 in 6
(1 change made)
. replace freq=.16*66 in 7
(1 change made)
. replace freq=.19*66 in 8
(1 change made)
```

```
. list
            col    agegroup       freq
   1.         1           1          5
   2.         1           2          9
   3.         1           3         19
   4.         1           4         33
   5.         2           1      27.72
   6.         2           2      15.18
   7.         2           3      10.56
   8.         2           4      12.54
```

Now use Stata's `expand` command to create a raw data set that has both observed values and expected values of the ordinal variable `agegroup`.

```
. expand =freq,clear
(122 observations created)
```

Separate the observed values of `agegroup` from the expected values with the following `generate` commands.

```
. generate x1=agegroup if col==1
(64 missing values generated)
. generate x2=agegroup if col==2
(66 missing values generated)
```

Finally, we can call upon Stata to compare the observed distribution against the expected distribution.

```
. ksmirnov x1=x2
Test: Equality of Distribution Functions (Kolmogorov-Smirnov)

Smaller group       D      P-value  Corrected
-----------------------------------------------
x1:              0.0000    1.000
x2:             -0.4441    0.000
Combined K-S:    0.4441    0.000     0.000
```

Note: The `onesamp` option to the `ksmirnov` command can be used if the expected theoretical distribution is specified by a Stata distribution function formula. The example in the Stata *Reference Manual* to test whether `x` is normally distributed with a mean of 4.571429 and a standard deviation of 3.45722 should read

```
. ksmirnov x = normprob((x-4.571429)/3.45722), onesamp
```

[See page 377 of the Reference Manual. The 'onesamp' was inadvertently left off in early printings, but in later printings, the error was corrected—Ed.]

The Kolmogorov–Smirnov Two Variable Test. The Kolmogorov–Smirnov two variable test can be employed when a dichotomous nominal variable separates the units of analysis into two groups and a second variable identifies the ranking of each unit of analysis on an ordinal scale. Consider this example from Sidney Siegel's classic text *Nonparametric Statistics for the Behavioral Sciences* (McGraw–Hill Book Company, 1956).

> In a study of correlates of authoritarian personality structure, one hypothesis was that persons in high authoritarianism would show a greater tendency to possess stereotypes about members of various national and ethnic groups than would those in low authoritarianism. This hypothesis was tested with a group of 98 randomly selected college women. Each subject was given 20 photographs and asked to "identify" those whose nationality she recognized, by matching the appropriate photograph with the name of the national group. Subjects were free to "identify" (by matching) as many or as few photographs as they wished. Since, unknown to the subjects, all photographs were of Mexican nationals—either candidates for the Mexican legislature or winners in a Mexican beauty contest—and since the matching list of 20 different national and ethnic groups did not include "Mexican," the number of photographs which any subject "identified" constituted an index of that subject's tendency to stereotype. (Siegel, pp. 132)

Authoritarianism was measured by a standardized test and subjects were grouped into "low" and "high" categories.

```
                NUMBER OF LOW AND HIGH AUTHORITARIANS
             "IDENTIFYING" VARIOUS NUMBERS OF PHOTOGRAPHS
        +----------------------+----------------------+
        |                      | Authoritarian score  |
        |            Number of |                      |
        | Row      photographs |    Low        High   |
        | no.       identified |     1           2    |
        +----------------------+----------------------+
        |   1            0-2   |    11           1    |
        |   2            3-5   |     7           3    |
        |   3            6-8   |     8           6    |
        |   4           9-11   |     3          12    |
        |   5          12-14   |     5          12    |
        |   6          15-17   |     5          14    |
        |   7          18-20   |     5           6    |
        +----------------------+----------------------+
        |            Total     |    44          54    |
        +----------------------+----------------------+
```

The computation proceeds by finding the maximum difference between the cumulative relative frequencies.

```
          NUMBER OF LOW AND HIGH AUTHORITARIANS "IDENTIFYING"
                   VARIOUS NUMBERS OF PHOTOGRAPHS:
              COMPUTATIONS FOR THE KOLMOGOROV-SMIRNOV TEST
   +----------------------+----------------------+------------+
   |                      | Authoritarian score  |            |
   |                      | (cumulative          |            |
   |                      | relative frequency)  |            |
   |            Number of |                      |            |
   | Row      photographs |    Low        High   |            |
   | no.       identified |     1           2    | Difference |
   +----------------------+----------------------+------------+
   |   1            0-2   |   0.250       0.018  |   0.232    |
   |   2            3-5   |   0.409       0.074  |   0.335    |
   |   3            6-8   |   0.591       0.185  |   0.406    |
   |   4           9-11   |   0.659       0.407  |   0.252    |
   |   5          12-14   |   0.773       0.630  |   0.143    |
   |   6          15-17   |   0.886       0.704  |   0.182    |
   |   7          18-20   |   1.000       1.000  |   0.000    |
   +----------------------+----------------------+------------+
```

The Kolmogorov–Smirnov D statistic equals 0.406, the maximum difference between the cumulative relative frequency distributions of the two groups. This indicates a statistically significant difference between the groups. "We conclude that women who score high on the authoritarian scale stereotype more ["identify" more photographs] than do women who score low on the scale." (Siegel, pp. 134)

The first step in using Stata is to input the crosstabulation. This is done by specifying each cell by column number, row number, and frequency.

```
        . input authval nphoto freq
               authval      nphoto       freq
        1.           1           1         11
        2.           1           2          7
        3.           1           3          8
        4.           1           4          3
        5.           1           5          5
        6.           1           6          5
        7.           1           7          5
        8.           2           1          1
        9.           2           2          3
       10.           2           3          6
       11.           2           4         12
       12.           2           5         12
       13.           2           6         14
       14.           2           7          6
```

Next, the raw data set is created by using Stata's expand command.

```
        . expand =freq,clear
        (84 observations created)
```

The values of the ordinal variable `nphoto` for the "low" group are separated from the "high" group with the following generate commands:

```
. generate x1=nphoto if authval==1
(54 missing values generated)

. generate x2=nphoto if authval==2
(44 missing values generated)
```

Now Stata's `ksmirnov` command will test the equality of the distributions, confirming the previous "hand" calculation:

```
. ksmirnov x1=x2

Test: Equality of Distribution Functions (Kolmogorov-Smirnov)

Smaller group        D       P-value  Corrected
----------------------------------------------
x1:                0.4057    0.000
x2:               -0.0025    1.000
Combined K-S:      0.4057    0.001    0.000
```

sqv1.1	Correction to logit regression extensions

Joseph Hilbe, Editor, STB, FAX 602-860-1446

It has been reported to me that use of `logiodd2` with a variable named "one" causes a problem. In fact, I used the names `one`, `good`, `depvar`, and `num` as working variables in my code, so use of any of these names would cause problems. I have fixed this. Please reinstall `extlogit.ado` from `\sqv1.1`.

srd1	How robust is robust regression?

Lawrence C. Hamilton, Department of Sociology, University of New Hampshire

[If you have installed either the January 1991 Support Disk or the CRC *materials from* STB-1 *or* STB-2, *you have the* `rreg` *described below.* `breg` *is provided in the* `\srd1` *directory on the* STB-2 *disk.—Ed.]*

The popularity of ordinary least squares (OLS) regression derives partly from its theoretical advantages given ideal data.[1] If errors are normally, independently, and identically distributed (normal *i.i.d.*), then OLS is more efficient than any other unbiased estimator. The flip side of this statement often gets overlooked: if errors are not normal, or not *i.i.d.*, then other unbiased estimators may outperform OLS. In fact, the efficiency of OLS degrades quickly in the face of heavy-tailed (outlier-prone) error distributions. Yet such distributions are common in many fields.

OLS tends to track outliers, fitting them at the expense of the rest of the sample. Over the long run, this leads to greater sample-to-sample variation or inefficiency when samples often contain outliers. Robust regression methods aim to achieve almost the efficiency of OLS with ideal data, and substantially better-than-OLS efficiency in non-ideal (e.g., not normal *i.i.d.*) situations. The *Stata News* (January 1991, 7(1):6) briefly described `rreg`, a Stata ado-file for one type of robust regression. Unlike OLS, "robust regression" is not a single, unified method. A broad range of robust estimators exist, with no strong consensus about which one works best. Since robust methods and the problems for which they are needed tend to be theoretically difficult, Monte Carlo methods play an important role in their evaluation.[2]

This article compares the performance of OLS and two robust regression procedures: `rreg` and a bounded-influence variant called `breg`. A more in-depth discussion, including background and examples, appears in *Regression with Graphics.*[3]

Two Robust Regression Programs

`rreg` works iteratively: it performs a regression, calculates case weights based on absolute residuals, and regresses again using these weights. Iterations stop when the maximum change in weights drops below a pre-specified tolerance (default tolerance= .01). Weights derive from one of two weight functions.

Huber weighting: Cases with small residuals receive weights of 1 (no downweighting), but cases with larger residuals get gradually lower weights. Let e_i represent the ith-case residual $Y_i - \mathbf{X}_i\mathbf{B}$ and MAD the median absolute deviation from the median residual,

$$\text{MAD} = \text{med}(|e_i - med\{e_i\}|)$$

The ith scaled residual u_i is

$$u_i = e_i/s$$

where s is a residual scale estimate; `rreg` employs $s = \text{MAD}/.6745$. Huber estimation finds case weights w_i:

$$w_i = \begin{cases} 1 & \text{if } |u_i| \leq c \\ c/|u_i| & \text{otherwise.} \end{cases}$$

c is a *tuning constant.* `rreg`'s Huber iterations employ $c = 1.345$, so downweighting begins with cases whose absolute residuals exceed about $2 \cdot \text{MAD}$.

Biweight: All cases with nonzero residuals receive some downweighting, according to the smoothly decreasing biweight function:

$$w_i = \begin{cases} [1 - (u_i/c)^2]^2 & \text{if } |u_i| \leq c \\ 0 & \text{otherwise.} \end{cases}$$

Very large residuals ($|u_i| \geq c$) result in zero weights—such severe outliers effectively drop out of the analysis.

`rreg`'s biweight iterations employ the tuning constant $c = 4.685$, so cases with absolute residuals of $7 \cdot \text{MAD}$ or more get zero weight (dropped).

The tuning constants 1.345 (Huber) and 4.685 (biweight) should give these robust procedures about 95% of the efficiency of OLS when applied to data with normally-distributed errors. Lower tuning constants would downweight outliers more drastically (but give up some Gaussian efficiency); higher tuning constants would make these estimators more like OLS.

Biweight estimation sometimes fails to converge or leads to multiple solutions, for which reason many analysts prefer Huber estimation. On the other hand, the biweight better protects against severe outliers. `rreg` attempts to exploit the strong points of both: it begins with OLS, switches to Huber weighting, and finishes with the biweight.[4] The `rreg` version described in the *Stata News* switches from Huber to biweight when the maximum change in weights equals five times the tolerance. Another option is to let Huber iterations continue until reaching tolerance, then perform a single biweight iteration. This second option works around the biweight's occasional instability, but tends to allow severe outliers to retain more influence. Though the two options may reach different solutions in any one sample, they produced virtually identical long-run behavior in the Monte Carlo experiment described below.

Like other robust M-estimators, `rreg` is designed for protection against wild errors or Y-outliers. X-outliers are its Achilles' heel. A case with unusual Y *and* X values may exert so much influence that `rreg` fails to spot it as an outlier. Tracking this outlier, `rreg` may even downweight other "innocent" cases instead—making the situation worse. Thus an analyst using M-estimators like `rreg` should carefully screen for possible leverage (X-outlier) problems before trusting the results.

Bounded-influence methods are another class of robust estimators, designed to cope with data containing both X- and Y-outliers. David Ruppert (personal communication) suggested a simple modification that turns `rreg` (or any other robust M-estimator) into a "quick-and-dirty" bounded-influence method.[5] Begin by performing OLS, and define c^H as the 90th sample percentile of h_i (leverage, or the ith hat matrix diagonal). Initial leverage-based weights, w_i^H, are

$$W_i^H = \begin{cases} 1 & \text{if } h_i \leq c^H \\ (c^H/h_i)^2 & \text{otherwise.} \end{cases}$$

Proceed with robust estimation as usual, but instead of applying just the Huber or biweight weights, w_i, at each step weight by $w_i w_i^H$. (Note, w_i changes with each iteration, whereas w_i^H remains constant.)

The automatic do-file `breg.ado`, a modified 1-biweight version of `rreg.ado`, carries out these calculations. `breg` provides some protection not only against wild errors in Y, but also (unlike `rreg`) against wild errors in X. Unfortunately, `breg` does not yet calculate valid standard errors and hypothesis tests. *The standard errors and tests it prints are wrong.* Asymptotically correct standard errors will involve further programming work.[6] As a substitute, inconvenient but attractive for its robustness, one might estimate `breg` standard errors by bootstrapping. Lack of standard errors does not detract from `breg`'s usefulness as a descriptive or diagnostic tool, for example to check whether leverage problems are distorting an OLS or `rreg` regression.

A Monte Carlo Experiment

Tables 1–3 summarize results from Monte Carlo simulations involving 1,000 samples of $n = 100$ cases each, generated according to three different models:

Model 1: fixed normal X; normal *i.i.d.* errors. This best-case model fits "the usual" assumptions made for OLS.

Model 2: fixed normal X; nonnormal but *i.i.d.* errors. The contaminated error distribution contains a proportion of large positive errors. Nonnormal errors are sometimes viewed as a "minor" violation of the usual assumptions, since they do not bias coefficient or standard error estimates.

Model 3: random nonnormal X; nonnormal and not *i.i.d.* errors. Large errors are more likely to coincide with extreme X values—a serious but realistic violation of the usual assumptions.

Error distributions in Models 2 and 3, and the X distribution in Model 3, are heavy-tailed and skewed. A good robust procedure should be able to handle skewed distributions, not just the symmetrical errors employed in many Monte Carlo studies.

Tables 1–3 give means, standard deviations, minima, and maxima of regression coefficients estimated four ways: by OLS; `rreg` with full biweight iterations; `rreg` with one biweight iteration; and `breg`. The tables repay careful study. For all three models, the true parameter values are

$$\text{Y-intercept } \beta_0 = 5 \text{ estimated by } b_0$$
$$\text{slope } \beta_1 = 1 \text{ estimated by } b_1$$

Mean estimated coefficients are close to these parameters, suggesting that all four regression methods provide unbiased estimates of these models' parameters.

As the Gauss–Markov Theorem predicts, OLS is the most efficient estimator with Model 1. `rreg`'s tuning constants supposedly make it about 95% as efficient as OLS, given normal errors. Results in Table 1 (relative efficiencies of 95–96%) suggest this tuning succeeds. Thus `rreg` performs almost as well as OLS, with ideal data. Monte Carlo results from full-biweight and 1-biweight versions of `rreg` were not really identical, but they appear so to three decimal places. The bounded-influence `breg` method performs notably worse than OLS or `rreg` with Model 1 (efficiencies 83–92%); it loses efficiency by needlessly downweighting cases from the X-distribution tails.

Given the heavy-tailed (but still *i.i.d.*) errors of Model 2, `rreg` far outperforms OLS and remains somewhat better than `breg`. Bounded-influence regression (`breg`) comes into its own with the "acid test" Model 3, where leverage problems begin to break down even `rreg` (and do much worse to OLS).

With Model 1, both OLS and `rreg` standard errors appear unbiased (as they should be): the mean estimated standard error resembles the standard deviation of the slope estimates. `rreg` slopes for Model 2 also have standard deviations close to the mean estimated standard errors; OLS results are less close, perhaps reflecting OLS' greater sensitivity to outliers.

Non-*i.i.d.* errors (Model 3) invalidate both OLS and `rreg` standard errors, as shown in Table 3. Though Tables 1–3 do not show this, `rreg` standard errors, based on asymptotic theory, also perform poorly in small samples (say, $n - K < 30$). With non-*i.i.d.* errors and/or small samples, bootstrapping may provide better standard error estimates.

Bootstrapping Robust Regression

Bootstrapping, or randomly resampling from the data at hand, promises to obtain maximum-likelihood standard error estimates under a wide variety of conditions—much wider than those covered by classical theory. A future article will discuss bootstrapping with Stata, and present further results. Here I just make some observations based on work done for *Regression with Graphics*.

Given well-behaved data, bootstrap (by *residual resampling*) standard deviations seem reasonably close to the standard errors estimated routinely by OLS or `rreg`. Furthermore, OLS tends to outperform `rreg` by margins similar to those seen in Table 1. With Y-outliers, but well-behaved X, bootstrap standard deviations indicate that `rreg` has the least sampling variation—corresponding to the situation we observed in Table 2. X-outliers give `breg` the advantage. Thus bootstrap results broadly confirm theoretical expectations and the Monte Carlo experiment of Tables 1-3. Bootstrapping adds the ability to explore which estimator works best *with the real data at hand.* It also provides a way to estimate standard errors for `breg`.

`rreg`'s standard errors derive from asymptotic (large-sample) theory. Small-sample behavior of `rreg` and most other robust estimators (unlike that of OLS) is not well understood. In my experience, bootstrap standard deviations begin to diverge sharply from `rreg` standard errors when $n - K < 30$, so this might be tentatively viewed as a point where sample sizes become too small to trust the asymptotic robust standard errors and tests. Monte Carlo simulations could investigate the question more thoroughly.

Residual resampling, like the theoretical standard errors, assumes *i.i.d.* errors. If this assumption seems implausible, standard errors may be estimated by another bootstrap approach called *data resampling*. Data resampling provides a last-resort path to standard errors, confidence intervals, and hypothesis tests for theoretically intractable problems.

Bootstrapping is a relatively new idea, with a lively and contentious literature. Some basic issues are not yet firmly resolved. Forming bootstrap confidence intervals, for instance, appears to be much less straightforward than initially supposed (and claimed in several texts).

Summary

The Monte Carlo experiment of Tables 1–3 illustrates strengths and weaknesses of each method:

1. Given fixed X and normal *i.i.d.* errors, OLS is the best unbiased estimator.[7]

2. `rreg` performs well when the data contain occasional wild errors (Y-outliers). Error distributions need not be normal or even symmetrical.

3. `breg` works better than `rreg` when the data contain both wild errors and X outliers (leverage points).

Using an insufficiently robust method risks worse trouble than using a too-robust method.

These guidelines are general. Which method works best with the data at hand? With bootstrapping, we can investigate the performance of different estimators applied to our own data and model. Bootstrapping also estimates standard errors, most useful when theory-based standard errors are untrustworthy due to false assumptions or small sample size, or unavailable due to computational complexity.

This article mentions only regression, but regression has many faces. To perform robust ANOVA or ANCOVA with `rreg`, simply create an appropriate set of dummy, slope dummy, or effect-coded X variables.[8] For example, a robust difference-of-means test, corresponding to the familiar two-sample t test, is achieved by regressing Y on a $\{0, 1\}$ dummy. Or suppose we just want one robust mean, for a variable called `energy`:

```
. generate zero=0
. rreg energy zero
```

Stata automatically drops the constant `zero` and prints a table in which the regression constant equals the robust mean of `energy`.

Robust methods protect against certain common data problems, and hence might be safer than OLS in the hands of naive analysts. Robust estimation is hardly foolproof, however. For example, `rreg` is susceptible to leverage just as OLS is. Neither `rreg` nor `breg` confers immunity to such bugaboos as nonlinearity, heteroscedasticity, autocorrelation, or multicollinearity (except by limiting the impact of occasional wild errors due to such problems). In short, robust methods cannot relieve the analyst of the need for careful diagnostic work, looking into and thinking about the results of any analysis. Analytical graphs (scatterplots, leverage plots, residual diagnostics, etc.) remain an indispensable aid to robust regression, as they are to any analysis.

Tables

Table 1: Monte Carlo Simulation Results, 1,000 artificial samples of $n = 100$ cases each, based on Model 1, $Y_i = 5 + 1 \cdot X_i + \epsilon_i$ where X is fixed, approximately $N(0, 1)$, and $\epsilon_i \sim N(0, 1)$.

Coefficient	Mean	Standard Deviation	Min	Max	Relative Efficiency[9]
OLS b0	4.995	.097	4.673	5.306	100%
rreg full-biweight b0	4.995	.099	4.685	5.308	96%
rreg 1-biweight b0	4.995	.099	4.686	5.308	96%
breg b0	4.995	.101	4.667	5.332	92%
OLS b1	.996	.104	.666	1.353	100%
rreg full-biweight b1	.996	.106	.646	1.354	95%
rreg 1-biweight b1	.996	.106	.646	1.354	95%
breg b1	.997	.114	.634	1.368	83%
OLS SE b1	.105	.010	.076	.134	
rreg full-biweight SE b1	.108	.011	.079	.141	
rreg 1-biweight SE b1	.108	.011	.079	.141	

Table 2: Monte Carlo Simulation Results, 1,000 artificial samples of $n = 100$ cases each, based on Model 2, $Y_i = 5 + 1 \cdot X_i + \epsilon_i$ where X is fixed, approximately $N(0, 1)$, and $\epsilon_i \sim N(0, 1)$ with probability .9 and $3(\chi_1^2 - 1)$ with probability .1.

Coefficient	Mean	Standard Deviation	Min	Max	Relative Efficiency[9]
OLS b0	4.990	.160	4.529	5.617	100%
rreg full-biweight b0	4.910	.115	4.501	5.279	193%
rreg 1-biweight b0	4.910	.115	4.501	5.279	193%
breg b0	4.911	.116	4.502	5.265	189%
OLS b1	.998	.168	.313	1.757	100%
rreg full-biweight b1	.996	.121	.610	1.402	191%
rreg 1-biweight b1	.996	.121	.610	1.402	191%
breg b1	.997	.129	.595	1.413	170%
OLS SE b1	.161	.052	.092	.539	
rreg full-biweight SE b1	.123	.013	.088	.167	
rreg 1-biweight SE b1	.123	.013	.088	.167	

Table 3: Monte Carlo Simulation Results, 1,000 artificial samples of $n = 100$ cases each, based on Model 3, $Y_i = 5 + 1 \cdot X_i + \epsilon_i$ where $X \sim N(0, 1)$, $e_i \sim N(0, 1)$ with probability .9 and $X \sim 3(\chi_1^2 - 1)$, $e_i \sim 3(\chi_1^2 - 1)$ with probability .1.

Coefficient	Mean	Standard Deviation	Min	Max	Relative Efficiency[9]
OLS b0	5.001	.163	4.576	5.718	100%
rreg full-biweight b0	4.920	.119	4.587	5.296	189%
rreg 1-biweight b0	4.920	.119	4.587	5.296	189%
breg b0	4.945	.114	4.614	5.293	204%
OLS b1	1.004	.225	.320	2.627	100%
rreg full-biweight b1	1.033	.144	.672	1.501	243%
rreg 1-biweight b1	1.033	.144	.672	1.501	243%
breg b1	1.054	.125	.639	1.476	322%
OLS SE b1	.107	.041	.029	.369	
rreg full-biweight SE b1	.082	.020	.019	.142	
rreg 1-biweight SE b1	.082	.020	.019	.142	

Notes

1. OLS encompasses not only regression as performed by Stata's `regress` command (without weighting), but also, as special cases, many common procedures based on arithmetic means (e.g., `oneway`, `anova`, `ttest`).

2. For an example Stata Monte Carlo program, see L. C. Hamilton (1991) ssi1: Monte Carlo simulation in STB-1.

3. L. C. Hamilton (forthcoming) *Regression with Graphics*. Pacific Grove, CA: Brooks/Cole.

4. This follows a suggestion made by G. Li in D. C. Hoaglin, F. Mosteller and J. W. Tukey, eds. 1985. *Exploring Data Tables, Trends, and Shapes*. New York: John Wiley & Sons. `rreg` estimates robust standard errors using the pseudovalues approach described by J. O. Street, R. J. Carroll, and D. Ruppert. 1988. "A note on computing robust regression estimates via iteratively reweighted least squares." *The American Statistician* May 42(2): 152–154.

5. This method is "quick and dirty" because it relies on the readily-available hat diagonals rather than some more robust but more difficult-to-obtain leverage measure. Better bounded-influence methods are described in D. G. Simpson, D. Ruppert, and R. J. Carroll. 1989. "One-step GM-estimates for regression with bounded influence and high breakdown-point," Technical Report 859, School of Operations Research and Industrial Engineering, Cornell University. For more about robust leverage measures, see P. J. Rousseeuw and B. C. Van Zomeren. 1990. "Unmasking multivariate outliers and leverage points," *Journal of the American Statistical Association* 85(411): 633–639, and subsequent papers.

6. `breg` is a Mallows-type estimator, so its standard errors follow from the usual theory of such estimators. For an approach that has been implemented in Minitab, see D. Wiens. 1990. "A Note on Computation of Robust, Bounded Influence Estimates in Regression," unpublished paper, University of Alberta.

7. Given *i.i.d.* (not necessarily normal) errors, OLS is BLUE: the best *linear* unbiased estimator. Given normal *i.i.d.* errors, OLS is the best unbiased estimator, linear or not. But with nonnormal errors, certain unbiased nonlinear estimators like `rreg` can outperform OLS.

8. Numerous texts, including *Regression with Graphics*, illustrate how to recast ANOVA as regression.

9. OLS variance as a percentage of estimator's variance. Values below 100% indicate worse-than-OLS performance; above 100% indicate better-than-OLS performance.

| srd2 | Test for multivariate normality |

Richard Goldstein, Qualitas, Brighton, MA, EMAIL goldst@harvarda.bitnet

The syntax of `multnorm` is

$$\texttt{multnorm } \textit{varlist } \big[\texttt{in } \textit{range}\big] \big[\texttt{if } \textit{exp}\big]$$

`multnorm` is a graphical procedure for examining multivariate normality, via diagnostics from a standard linear regression. Thus, you must enter at least two variables; do not enter any options other than `in` and/or `if`. Cases with missing values on any variable in *varlist* are dropped prior to producing the new variables and the graph.

This is a "test" of multivariate normality (actually a graph) taken from J. Stevens (1986) *Applied Multivariate Statistics for the Social Sciences*, Hillsdale: L. Erlbaum Assoc., Publishers, pp. 207–212, and from B. Thompson (1990), "MULTINOR: A Fortran Program that Assists in Evaluating Multi-variate Normality," *Educational and Psychological Measurement*, 50: 845–8. Note that Stevens contains some typos and approximations; although Thompson does not mention any problem with Stevens' calculation of Mahalanobis Distance, Thompson's graph agrees with that calculated here for Stevens' data and *not* with Stevens' calculations nor his graph. Note also that the formula for Mahalanobis Distance used here is considered "inappropriate" for use as a measure of leverage by P. F. Velleman and R. E. Welsch (1981), "Efficient Computing of Regression Diagnostics," *The American Statistician*, 35: 234–242; the variable MD2 used and reported here, and used by Thompson and, apparently, Stevens, appears as equation 29 in Velleman & Welsch.

If the variables are multivariate normal, then the graph will approximate a 45-degree line.

Note that the calculations are relatively slow.

The Stevens data, along with the ado and help files, are included in the \srd2 directory of the STB-2 disk as `stevens.dta`. An example of using this command is provided in the on-line help.

| srd3 | One-Step Welsch bounded-influence estimator |

Richard Goldstein, Qualitas, Brighton, MA, EMAIL goldst@harvarda.bitnet

The syntax of `bound` is

$$\texttt{bound } \textit{depend\_var varlist } \big[\texttt{in } \textit{range}\big] \big[\texttt{if } \textit{exp}\big]$$

`bound` estimates a one-step Welsch bounded-influence regression. The first part of the output is an ordinary least squares regression. Next a number of regression diagnostics are computed. Finally, "dffits" is used to weight the data and estimate a one-step Welsch bounded-influence regression. This is not the full Krasker-Welsch bounded-influence estimator.

This is suggested in R. E. Welsch (1980), "Regression Sensitivity Analysis and Bounded-Influence Estimation," in *Evaluation of Econometric Models*, ed. by J. Kmenta and J. B. Ramsey, New York: Academic Press, pp. 153–167; Welsch claims that the "cutoff of 0.34 is chosen for approximately 95% asymptotic efficiency." (p. 165)

Between the standard and one-step regression results, a list of cases appears with values on a number of diagnostics, including hat, studentized residuals, dffits, dfbeta on the first of the right-hand-side variables, Cook's distance, the covariance ratio, the likelihood distance and a probability for that distance. Only cases that cross the "rule-of-thumb" cutoff for any *one* of these are shown. Above the list, some of the cutoffs for that data set are presented. No cutoffs are shown for Cook's distance (I just use 1.0) or for the studentized residual (I just use 2.0). The cutoff for hat is $3p/n$ not the BKW suggestion of $2p/n$, as I have found this to be more informative in my own work. I have not found the covariance ratio or the likelihood distance to be worth very much, but have left them in case others find them helpful.

Do not use any standard regression options, or `test`, as they interfere with the purpose of this file. An example of using this command is provided in the on-line help.

srd4	Test for general specification error in linear regression

Richard Goldstein, Qualitas, Brighton, MA, EMAIL goldst@harvarda.bitnet

The syntax of `pswdiff` is

$$\texttt{pswdiff } \textit{varlist } \big[\texttt{in } \textit{range}\big] \big[\texttt{if } \textit{exp}\big]$$

`pswdiff` is a test for general (i.e., non-specific) specification error in a standard linear regression; it is implemented as a test for omitted variables, via added variables as a function of lags and leads of the included right-hand-side variables. *Do not* include a constant, or seasonal dummies, or polynomial terms in your model; do not include a lagged version of the left-hand-side variable on the right-hand-side. A version of this test including a lagged dependent variable is provided in the citation below.

The reported regression itself is of little (no) interest; what is of interest is the joint test of significance reported after the regression: if significant, you have left out at least one important variable or you have the wrong functional form; if not significant, then you may not have one of these problems.

The PSW difference test for specification error is described in R. Davidson, L. Godfrey, and J. G. MacKinnon, (1985), "A Simplified Version of the Differencing Test," *International Economic Review*, 26, pp. 639–47. This is a general test for model misspecification—applicable to time-series data. It is equivalent to a test for omitted variables, and is accomplished via adding the omitted variables which are defined as the sum of the lagged and leaded values of the variables. An example of using this command is provided in the on-line help.

srd5	Ramsey test for heteroscedasticity and omitted variables

Richard Goldstein, Qualitas, Brighton, MA, EMAIL goldst@harvarda.bitnet

The syntax of `ramsey` is

$$\texttt{ramsey } \textit{varlist } \big[\texttt{in } \textit{range}\big] \big[\texttt{if } \textit{exp}\big]$$

`ramsey` includes two tests for specification error: the first is a test of heteroscedasticity; the second a test for possible omitted variables. There are two versions of this last test. In all cases, your real interest should be in the results of a joint test of significance of the added variables that appear after the regression; the regression itself is of little or no interest. If significant, you have a problem! The second version of the omitted variables test is more powerful than the first version unless you have dummy variables in the regression—in that case, it will not be meaningful—this will announce itself since a number of variables will be dropped by Stata in the regression. See J. G. Thursby and P. Schmidt (1977), "Some Properties of Tests for Specification Error in a Linear Regression Model," *JASA* 72: 635–41. `ramsey` limits these powers to non-dummy variables, which it identifies by the storage type of the variable. Variables stored as `byte`s are assumed to be dummies, the rest nondummies. (Thus, be sure that nondummy variables are not stored as `byte` if you want to use their powers in the test; on the other hand, ensure that dummy variables do have a type of `byte`.)

The first regression output is the standard regression being tested.

It can sometimes happen that even though there are no dummy variables, the second version of the second test is not meaningful as most of the 6 constraints will be dropped! This will tend to happen in quadratic or other polynomial models. In general for polynomial models, the `pswdiff.ado` file (see *srd4*) will be more informative than will the Ramsey RESET test. An example of using this command is provided in the on-line help.

srd6	A randomization test for the equality of two groups

Richard Goldstein, Qualitas, Brighton, MA, EMAIL goldst@harvarda.bitnet

The syntax of `urnmodel` is

$$\texttt{urnmodel } \textit{varlist } \big[\texttt{in } \textit{range}\big] \big[\texttt{if } \textit{exp}\big]$$

`urnmodel` uses an approximation to a randomization test on the residuals of a standard linear regression to test the equality of two groups (e.g., males and females)—*do not* include a dummy variable for these groups in your regression. This first variable in *varlist* should be the group variable to be tested—the code expects this variable to be coded as a 0-1 dummy variable. The second variable in *varlist* is the dependent variable for the regression, and the remaining variables are the right-hand-side variables for the regression.

This model is discussed in some detail in B. Levin and H. Robbins (1983), "Urn Models for Regression Analysis, with Applications to Employment Discrimination Studies," *Law and Contemporary Problems*, 46, pp. 247–67, and more briefly in

M. O. Finkelstein and B. Levin (1990), *Statistics for Lawyers*, New York: Springer-Verlag, esp. at pp. 399–402. A strata-oriented extension of this model is described on pp. 253–5 of Levin and Robbins.

Note that this model should be "less significant" than a model that (1) includes a sex coefficient or (2) includes sex by variable interactions, since "To the extent that sex is a factor in determining salary, and is correlated with the productivity [sic] factors, its effect is assigned to those productivity [sic] factors" (F&L, p. 402). Among other things, this means that this procedure will generally have lower power than either of the above two procedures. Both citations show the algebraic relationship between the z test calculated here for the urn model and the t test resulting from a regression with a dummy variable for sex.

If you have available a randomization t test procedure, that will be more accurate than the approximation used here: write out the residuals with the grouping variable to the other package and use the exact randomization test there. If they only have an approximate randomization test, you should probably use both urn model and the approximate test. The approximation used here is based on the normal distribution. An example of using this command is provided in the on-line help.

ssa1.1	Menu interface to life-table command

William Gould, CRC, FAX 213-393-7551

In *crc8*, we offer a rewrite of `lftbl` presented in *ssa1* by Henry Krakauer and John Stewart, Office of Research, Health Care Financing Administration. Our rewrite, in fitting with standard Stata syntax, is command, not menu, driven. Upon examination of Krakauer's and Stewart's code, it was clear they went to a lot of trouble to produce a menu front-end. User's wanting a menu front-end can use `ltablem`, to be found in `\ssa1.1` on the STB-2 disk. Type '`ltablem`' and follow the instructions.