STATA
TECHNICAL
BULLETIN

November 1996
STB-34

A publication to promote communication among Stata users

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

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

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

The *Stata Technical Bulletin* (ISSN 1097-8879) is published six times per year by Stata Corporation. Stata is a registered trademark of Stata Corporation.

Contents of this issue

page

dt3.1 An updated utility to convert EpiInfo datasets

R. Mark Esman, Stata Corp., FAX (409) 696-4601, tech@stata.com

This is an updated version of `epi2dct` which appeared as *dt3* in STB-32. `epi2dct` is a stand-alone DOS executable program written to convert data produced with EpiInfo into a format readable by Stata. This version offers improved reading of records spanning multiple lines. `epi2dct` reads EpiInfo's record (`.rec`) files and then writes out a file in Stata's dictionary (`.dct`) file format. The resulting dictionary file can then be read into Stata using the `infile` command on any platform.

For those users who do not use DOS or Windows, the source code is included.

Since the first version of this program was released in STB-32, several Stata users have contacted me with difficulties reading record files with a large number of variables. In particular, files in which the data span multiple lines. I have tried to address most of the reported problems with this version. One particular problem that should be mentioned is the case where a string variable(s) contain an exclamation point (`!`). EpiInfo uses this character as a line delimiter and therefore must be stripped from the data during the conversion process.

Again, I would appreciate feedback from those of you that use `epi2dct` and/or EpiInfo. I have had the opportunity to try converting some "real world" record files with very favorable results. In many cases, this program will be useful to eliminate most of the mundane work, but the resulting dictionary files may need minor editing. As it stands, the program knows only two data types: strings and floats. After the dictionary is created, you may edit the datatypes to include the appropriate cast. If, in the future, there are any updates made, I will include them on the Stata Web site: http://www.stata.com.

Syntax

`epi2dct` is executed by typing the following from a DOS prompt:

`C:> epi2dct` *infilename outfilename*

or shelling out from the Stata dot prompt in Stata for Windows or Stata for DOS:

`.!epi2dct` *infilename outfilename*

The *infilename* is the name of the EpiInfo format (`.rec`) record file used to read data into the `epi2dct` program; *outfilename* is the file in which the Stata format (`.dct`) dictionary file will be written. It should be noted that `epi2dct` will not assume the proper file extensions; you must remember to add the correct extensions manually. Typing `epi2dct` alone at the DOS prompt will display a syntax diagram and short help file.

Options

There are no options associated with `epi2dct`.

Example

The following is an example EpiInfo data file which was created to illustrate how the `epi2dct` program converts data. The file was created using a sample questionnaire in EpiInfo 6.0 and converted with the `epi2dct` program.

```
16 1
PERSONRECO   34   2  30  47   2   0   0 112 Person Record
ILLUSTRATI    5   4  30  72   4   0   0 112
RECORDSBYH    5   5  30  75   5   0   0 112
IDENTIFIER    5   6  30  77   6   0   0 112
VIADEFINED    5   7  30  76   7   0   0 112
PASSEDVALU    5   8  30  72   8   0   0 112
ENTEREDHER    5   9  30  18   9   0   0 112
PERSONID     21  11  30  30  11  12   5 112 PersonID
HOUSEID      44  11  30  52  11   6   5 112 HouseID
NAME         21  13  30  26  13   1  33 112 Name
AGE          26  15  30  31  15   0   3 112 Age
SEX          38  15  30  42  15   3   1 112 Sex
ILL          48  15  30  53  15   5   1 112 Ill
ADDRESS      21  17  30  29  17   1  32 112 Address
CITY         21  19  30  26  19   1  22 112 City
STATE        50  19  30  56  19   3   2 112 State
```

```
  1 1111Santa Q. Claus                    99MN100 North Pole Circle         !
North Pole            AK!
  2 2222Easter A. Bunny                   30FN555 Hare Lane                 !
Everywhere            TX!
  3 3333Tooth C. Fairy                    45FY32 Bicuspid Blvd.             !
Molar                 CO!
  4 4444Frosty T. Snowman                 55MY5432 Corncobb Ct.             !
Winter Wonderland     MT!
```

which we can convert to a Stata dictionary file using

```
C:> epi2dct sample.rec sample.dct
```

The created dictionary file is

```
dictionary {
        str5 personid %5s "PersonID"
        _column(6)
        str5 houseid %5s "HouseID"
        _column(11)
        str33 name %33s "Name"
        _column(44)
        age %3f "Age"
        str1 sex %1s "Sex"
        _column(48)
        str1 ill %1s "Ill"
        _column(49)
        str32 address %32s "Address"
        _column(81)
        str22 city %22s "City"
        _column(103)
        str2 state %2s "State"
        _column(105)
}
    1 1111Santa Q. Claus                    99MN100 North Pole Circle           North Pole            AK
    2 2222Easter A. Bunny                   30FN555 Hare Lane                   Everywhere            TX
    3 3333Tooth C. Fairy                    45FY32 Bicuspid Blvd.               Molar                 CO
    4 4444Frosty T. Snowman                 55MY5432 Corncobb Ct.               Winter Wonderland     MT
```

The dictionary file can then be read into Stata using the `infile` command:

```
. infile using sample.dct
```

Reference

Esman, R. M. 1996. dt3: Reading EpiInfo datasets into Stata. *Stata Technical Bulletin* 32: 9–10.

gr20 | Low-level graphics in data coordinates

H. Joseph Newton, Texas A&M University, FAX (409) 845-3144, jnewton@stat.tamu.edu
James W. Hardin, Stata Corp., FAX (409) 696-4601, tech@stata.com

With the `gph` commands in Stata 5.0, programmers are now able to produce highly customized graphics. These commands allow one to open a graphics window in which points, lines, text, boxes, and so on can be drawn. One particularly important use of the commands is in adding graphical components to graphs produced by the `graph` command. In this insert we describe modified versions of the `gph` commands which allow a programmer to add graphical elements to an existing graph in terms of the data coordinates of the graph rather than in terms of screen coordinates as the `gph` commands require. We illustrate the use of the new programs with three examples.

Background

Stata thinks of a graphics window as being composed of a matrix of rows and columns of plotting positions with a particular position in the window being denoted by (r, c) where r and c denote the row and column of the position, respectively. The upper left and lower right corners of the window are positions $(0, 0)$ and $(23063, 32000)$, respectively. When a program calls the `graph` command, Stata places the bounding box of the graph and the data region of the graph into the global macros `S_G1` and `S_G2`, respectively (see [R] **gph** for details), as well as placing into `_result(5)` through `_result(8)` the factors needed to convert data coordinates to screen coordinates. If we denote these four factors by a_y, b_y, a_x, and b_x, then the screen coordinates (r, c) of a data coordinate having horizontal and vertical values x and y are given by

$$c = a_x x + b_x, \qquad r = a_y y + b_y$$

Syntax

`gphsave`

`gphdt clear` y_1 x_1 y_2 x_2

`gphdt text` y x $\#_{rotation}$ $\#_{alignment}$ *text*

`gphdt vtext` *varname*$_y$ *varname*$_x$ *varname*$_{str}$ [`if` *exp*] [`in` *range*]

`gphdt line` y_1 x_1 y_2 x_2

`gphdt vline` *varname*$_y$ *varname*$_x$ [`if` *exp*] [`in` *range*]

`gphdt vpoly` *varname*$_{y_1}$ *varname*$_{x_1}$ *varname*$_{y_2}$ *varname*$_{x_2}$ $\cdots$ *varname*$_{y_p}$ *varname*$_{x_p}$
[`if` *exp*] [`in` *range*]

`gphdt box` y_1 x_1 y_2 x_2 $\#_{shade}$

`gphdt point` y x $\#_{\Delta c}$ $\#_{symbol}$

`gphdt vpoint` *varname*$_y$ *varname*$_x$ [*varname*$_{\Delta c}$ *varname*$_{symbol}$] [`if` *exp*] [`in` *range*]
[`, size(`$\#_{\Delta c}$`) symbol(`$\#_{symbol}$`)`]

The gphsave command

For the `gphdt` commands to work properly, one must call `gphsave` immediately after calling `graph`. This puts the elements of `_result(5)` through `_result(8)` into the global macros `GPH_ay`, `GPH_by`, `GPH_ax`, and `GPH_bx` for the `gphdt` commands to use.

The gphdt commands

In this section we refer to a data coordinate as (x, y) where x and y are the horizontal and vertical coordinates of the point. In the arguments of the `gphdt` commands, we have preserved Stata's usual practice of putting the vertical position first.

`gphdt clear` y_1 x_1 y_2 x_2 clears the rectangle having opposite corners (x_1, y_1) and (x_2, y_2).

`gphdt text` y x $\#_{rotation}$ $\#_{alignment}$ *text* displays *text* at data coordinate (x, y). See [R] **gph** for information about the rotation and alignment arguments.

`gphdt vtext` *varname*$_y$ *varname*$_x$ *varname*$_{str}$ [`if` *exp*] [`in` *range*] displays N centered lines of horizontal text where the location and text for the lines are contained in the three variables.

`gphdt line` y_1 x_1 y_2 x_2 draws a line from data coordinate (x_1, y_1) to (x_2, y_2).

`gphdt vline` *varname*$_y$ *varname*$_x$ [`if` *exp*] [`in` *range*] draws a series of connected lines where the consecutive data coordinates are contained in the two variables.

`gphdt vpoly` *varname*$_{y_1}$ *varname*$_{x_1}$ *varname*$_{y_2}$ *varname*$_{x_2}$ $\cdots$ *varname*$_{y_p}$ *varname*$_{x_p}$ [`if` *exp*] [`in` *range*] draws a series of connected lines for each observation in the input variables.

`gphdt box` y_1 x_1 y_2 x_2 $\#_{shade}$ draws a rectangle having opposite corners having data coordinates (x_1, y_1) and (x_2, y_2). The shading argument must be between 0 and 5 where the shading gets darker from 0 to 4, and 5 means there is no shading.

`gphdt point` y x $\#_{\Delta c}$ $\#_{symbol}$ displays a point at data coordinate (x, y). See [R] **gph** for information on the size and symbol arguments.

`gphdt vpoint` *varname*$_y$ *varname*$_x$ [*varname*$_{\Delta c}$ *varname*$_{symbol}$] [`if` *exp*] [`in` *range*] [`, size(`$\#_{\Delta c}$`) symbol(`$\#_{symbol}$`)`] displays points at the data coordinates contained in the variables. See [R] **gph** for information on the size and symbol arguments.

A simple linear regression example

A standard example of adding graphic components to a plot is to add the least squares regression line to a scatterplot as well as line segments showing vertical deviations of observed points to the line. Here is a simple program called **slrplot** for doing this for a given dependent variable **y** and independent variable **x**. The program uses **gphdt vpoly** to draw the line segments and **gphdt line** to draw the regression line.

```
program define slrplot
        version 5.0
        local y "`1´"
        local x "`2´"
        tempvar yhat
        quietly {
                gph open
                graph `y´ `x´, xlab ylab
                gphsave
                regress `y´ `x´
                predict `yhat´
                gphdt vpoly `y´ `x´ `yhat´ `x´
                sum `x´
                local x1 = _result(5)
                local x2 = _result(6)
                local y1 = _b[_cons] + _b[`x´]*`x1´
                local y2 = _b[_cons] + _b[`x´]*`x2´
                gphdt line `y1´ `x1´ `y2´ `x2´
                gph close
        }
end
```

In Figure 1, we show the result of using **slrplot** for regressing **price** on **mpg** for Stata's **auto.dta**.

```
. slrplot price mpg
```

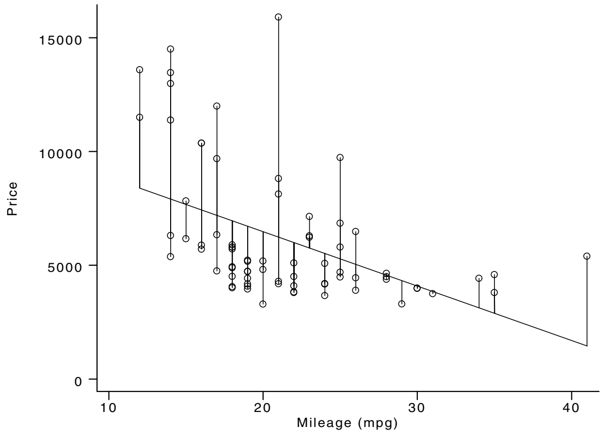


Figure 1.

In Figure 2, we show the result for a sample of size 30 from a bivariate normal population having means 0, variances 1, and correlation coefficient 0.7. We generated the data by

```
. set obs 30
. gen e1 = invnorm(uniform())
. gen e2 = invnorm(uniform())
. local rho = 0.7
. gen x = e1
. gen y = `rho´ * e1 + sqrt(1 - `rho´ * `rho´) * e2
. slrplot y x
```

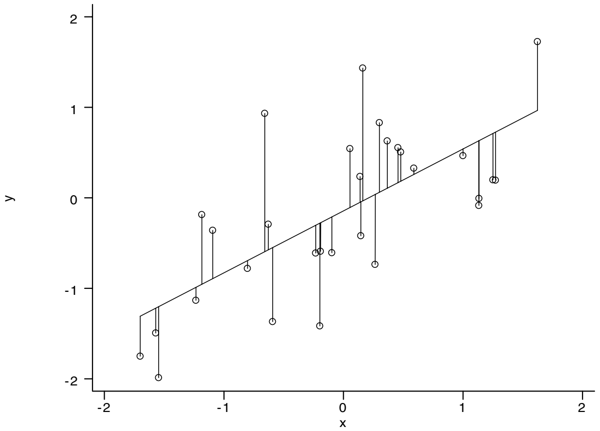


Figure 2.

Illustrating the rejection method of random number generation

If it is hard to generate realizations from a random variable X having pdf f and we can find another pdf g defined over the same range as f for which it is easy to generate realizations and we can find a constant $c > 1$ such that

$$h(x) = cg(x) \geq f(x) \qquad \text{for all } x$$

then we can generate a realization X from f via the following algorithm, called the rejection or acceptance-rejection method:

1. Generate Y from g.
2. Let $Z = Uh(Y)$ where U is $U(0,1)$ independent of Y.
3. Let $X = Y$ if $Z < f(Y)$, otherwise go back to 1.

The probability that a point will be rejected is the area between h and f divided by the area under h, that is,

$$\Pr(\text{rejection}) = \frac{\int_{-\infty}^{\infty}[h(x) - f(x)]dx}{\int_{-\infty}^{\infty} h(x)dx} = 1 - \frac{1}{c}$$

and thus we should try to find a c that is as small as possible. If we know that we can find an “enveloping function” g, then we can find the “best” value of c by finding

$$c = \max_x \left(\frac{f(x)}{g(x)} \right)$$

since we want $cg(x) \geq f(x)$.

We consider using the standard Cauchy pdf

$$g(x) = \frac{1}{\pi} \frac{1}{1 + x^2}, \quad -\infty < x < \infty$$

as the enveloping function for the standard normal pdf

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, \quad -\infty < x < \infty$$

which means we need to find the maximum of

$$q(x) = \frac{f(x)}{g(x)} = \sqrt{\frac{2}{\pi}} \frac{e^{x^2/2}}{1 + x^2}$$

In Figure 3, we have used `graph` to plot the function q and then used `gphdt line` and `gphdt text` to show where the maximum value $c = \sqrt{2\pi}e^{-1/2} \doteq 1.52$ occurs.

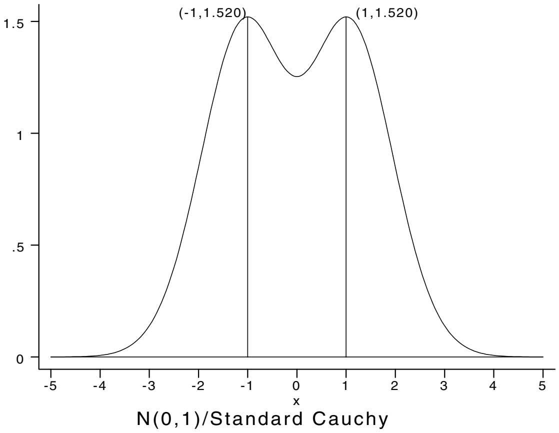


Figure 3.

In Figure 4, we have used **graph** to draw $h(x) = cg(x)$, then used **gphdt vline** to add the plot of $f(x)$, and finally **gphdt vpoly** to shade in the rejection region, that is, the area between the two curves.

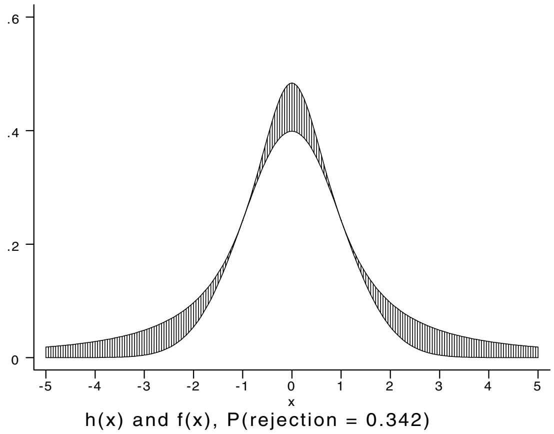


Figure 4.

Here is the program that was used to produce these two figures. Notice that the final graphs were saved by using the **saving** option on the **gph open** commands.

```
program define tryrej
        version 5.0

        quietly {

                gph open, saving(tryrej1.gph,replace)

                set obs 201
                gen x = 5*(_n-101)/100
                gen y = sqrt(_pi/2)*exp(-x^2/2)*(1+x^2)

                graph y x, c(l) s(i) xlab(-5,-4,-3,-2,-1,0,1,2,3,4,5) ylab /*
                */ title("N(0,1)/Standard Cauchy")
                gphsave

                local c = sqrt(2*_pi)*exp(-.5)
                local dc : display %5.3f `c´
                local oc : display %5.3f 1-(1/`c´)

                gphdt line `c´ -1 0 -1
                gphdt line `c´  1 0  1
                gphdt line  0  -5 0  5

                gphdt text `c´  1 0 -1 (1,`dc´)
                gphdt text `c´ -1 0  1 (-1,`dc´)

                gph close
                gph open, saving(tryrej2.gph,replace)
        *   Graph h(x)=c*g(x):
                gen h = `c´/(_pi*(1+x^2))
                graph h x, c(l) s(i) xlab(-5,-4,-3,-2,-1,0,1,2,3,4,5) ylab /*
                */ title("h(x) and f(x), P(rejection = `oc´)")
                gphsave
        *   Add N(0,1) curve and shade rejection region:
                replace y = exp(-x^2/2)/sqrt(2*_pi)
                gphdt vline y x
                gphdt vpoly y x h x
                gph close
        }
end
```

A simple mapping example

The file **census90.dta** contains variables **x** and **y** which are the longitude and latitude of the centroids of the 582 census tracts in Harris County, Texas in the 1990 census, as well the variable **kids** which contains the number of children five years old or younger in the 582 tracts. The file also contains variables called **lgtude** and **lttude** which are a set of coordinates for the boundary of Harris County (observations $1-47$) as well as eight other sets of observations, each of which is part of the system of roads in the County. In Figure 5 we have plotted the locations of the 582 centroids as well as the county boundary and main roads in the county. The **graph** function was used to set up the axes and plot the centroids, while **gphdt vline** was used nine times to get the boundary and roads. The program also uses the **bbox** option on **graph** and **gph text** to put a title above the graph.

Here is the program that was used to produce Figure 5.

```
program define plotrds2
        version 5.0
        use census90
        quietly {
                gph open, saving(plotrds2.gph,replace)
                graph x y, s(.) xlab(-96,-95.6,-95.2,-94.8) /*
                */  ylab(29.5,29.6,29.7,29.8,29.9,30,30.1,30.2) /*
                */  bbox(3000,0,23000,30000,850,400,0)
                gphsave
                gph text 1000 16000 0 0 1990 Census Tracts in Harris County, Texas
                gphdt vline lttude lgtude in 1/47
                gphdt vline lttude lgtude in 48/58
                gphdt vline lttude lgtude in 59/71
                gphdt vline lttude lgtude in 72/82
                gphdt vline lttude lgtude in 83/90
                gphdt vline lttude lgtude in 91/96
                gphdt vline lttude lgtude in 97/107
                gphdt vline lttude lgtude in 108/147
                gphdt vline lttude lgtude in 148/153
                gph close
        }
end
```

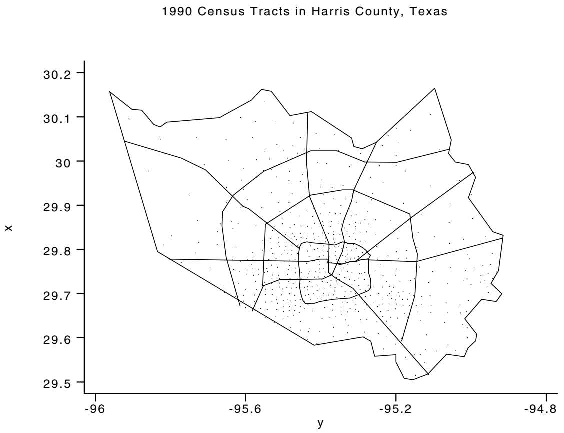


Figure 5.

gr21 Flexible axis scaling

Patrick Royston, Royal Postgraduate Medical School, UK, proyston@rpms.ac.uk

Stata's standard graphs allow linear and logarithmic axis scaling, but that's all. Sometimes you want other scales. tgraph is a utility to produce graphs with almost any scaling you like.

Consider Figure 1. This shows a scatterplot of mpg (miles per gallon) against displ (displacement, cu. in.) for the 74 cars in the ubiquitous automobile dataset auto.dta supplied with Stata. Clearly the relationship is quite nonlinear, but fractional polynomial analysis using Stata's fracpoly command reveals a simple transformation (x= displ^{-2}) which makes the relationship between mpg and x very nearly a straight line. In some applications, you might want to plot mpg against displ on the special scale (inverse square) for displ which gives such a straight line.

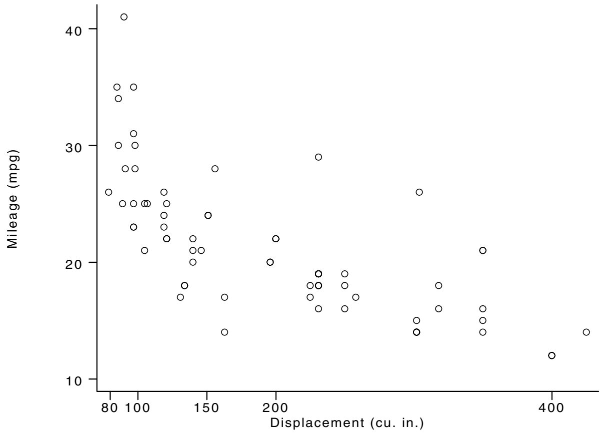


Figure 1.

In Figure 1, I have labelled the displacement axis with the unequally spaced numbers 80, 100, 150, 200, 400 cu. in. Figure 2 shows the same labels applied to the new scale in which the relationship is linear. One is actually plotting mpg against x, but keeping the original labels for displ. Notice how the label spacing, which looks quite odd in Figure 1, is appropriate for the new scale. Analogously, one often labels log scales with increments corresponding to powers of 10.

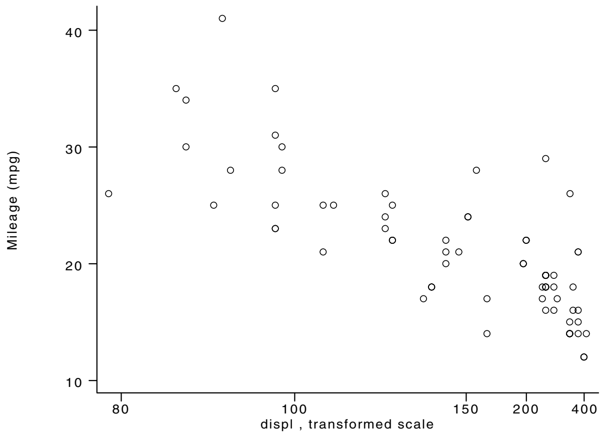


Figure 2.

Figures 1 and 2 were drawn using graph and tgraph as follows:

```
. graph mpg displ, xlabel(80,100,150,200,400) ylabel
. tgraph mpg displ, xlabel(80,100,150,200,400) xtrans(-1/@^2) ylabel
```

The first statement just produces Figure 1 and is standard. The second uses tgraph to create a plot with the x-axis transformed according to the function $-1/x^2$. The @ is a placeholder. The reason why the transformation is $-1/x^2$ rather than $1/x^2$ is that use of $1/x^2$ reverses the ordering of the x-axis labels. Sometimes you might want to do this, but not here.

You can if you wish transform both axes; you just define suitable labels for each axis and suitable transformations `xtrans` and `ytrans`. All y-variables are transformed in the same way.

Syntax

The syntax of `tgraph` is

`tgraph` *yvarlist xvar* [`if` *exp*] [`in` *range*] [`,` `xtrans(`*x_transf*`)` `xlabel(`*xlablist*`)`
`ytrans(`*y_transf*`)` `ylabel(`*ylablist*`)` *graph_ options*]

Options

`xtrans(`*x_transf*`)` defines the x-axis transformation; i.e., a function to be applied to *xvar*. The argument of the function is represented by `@`. Square brackets must be used in place of round parentheses; otherwise, standard Stata expression syntax is used. If you specify `xtrans()`, then you must also specify `xlabel()`.

`ytrans(`*y_transf*`)` defines the y-axis transformation; i.e., a function to be applied to each of the variables in *yvarlist*. The rules for defining the function are the same as those for `xtrans()`. If you specify `ytrans()`, then you must also specify `ylabel()`.

`xlabel(`*xlablist*`)` and `ylabel(`*ylablist*`)` are the required axis labels.

graph_options are any of Stata's `graph, twoway` options except for `xlabel()` and `ylabel()`.

Remarks

The principle behind `tgraph` is that it creates value labels for the transformed variable(s) which "point to" the labels you specified in `xlabel()` or `ylabel()` on the original scale of *xvar* or *yvarlist*. Stata automatically uses these value labels in the plot, even though you are plotting "strange" transformed variable(s) which you never see, and which `tgraph` does not preserve.

The only extra point worth noting is that in defining the transformations, any Stata expression may be used except that round parentheses should be replaced with square brackets. For example, to get a square root x-axis, you would specify `xtrans(sqrt[@])`, rather than `xtrans(sqrt(@))` which would result in a syntax error.

ip13 Maximum-likelihood estimation using the ml command

William Gould, Stata Corp., FAX (409) 696-4601, wgould@stata.com

The purpose of this insert is to take away some of the mystery of using Stata's `ml` command. `ml` allows you to specify your own likelihood function and obtain the corresponding maximum-likelihood estimates.

In outline, you (1) write a Stata program to evaluate the likelihood function, (2) give an `eq` command to define the model to be estimated, and (3) give a sequence of `ml` commands to obtain the solution.

`ml` provides four alternative methods for obtaining the maximum-likelihood estimates:

1. linear form (`lf`),
2. derivative-free (`deriv0`),
3. user-supplied first derivatives (`deriv1`),
4. user-supplied first and second derivatives (`deriv2`).

The form of the program you write for evaluating the likelihood function depends on the chosen method, but they are each variations on the same theme.

In this article, I will discuss the linear-form (`lf`) method exclusively. This method is the easiest to use and is almost as fast as the user-supplied first and second derivatives (`deriv2`) method.

The linear-form restrictions

Not all likelihood functions can be written in the `lf`-restricted way. The `lf` method places two restrictions on the form of the likelihood function:

$$\ln L(\mathbf{b};\mathbf{y},\mathbf{X}) = \sum_{j=1}^{N} \ln l(\mathbf{b};y_j,\mathbf{x}_j)$$

$$\ln l(\mathbf{b};y_j,\mathbf{x}_j) = \ln l(\mathbf{x}_j\mathbf{b};y_j)$$

The first restriction is that the log-likelihood can be expressed as a sum of the log-likelihoods for each observation; i.e., the observations are independent. Programmingwise, it means that you write a program not to calculate $\ln L$, the overall log-likelihood function; you write a program to calculate $\ln l$, the log-likelihood of an individual observation. `ml` itself will compute $\ln L$ by summing the individual components.

> **Aside:** Obtaining likelihood values observation-by-observation allows `ml` to better monitor the convergence and feasibility of $\mathbf{b}$. It is not uncommon in likelihood problems to choose, along the way, a $\mathbf{b}$ that results in infeasible values of $\ln l$ for some of the observations and, in forming sums, it is too easy to sum across such observations and treat their contribution as 0. `ml` watches for this problem.

The second restriction, $\ln l(\mathbf{b};y_j,\mathbf{x}_j) = \ln l(\mathbf{x}_j\mathbf{b};y_j)$, is the linear-form restriction. It states that only the inner product $\mathbf{x}_j\mathbf{b}$ enters the likelihood function. (Note that we write $\mathbf{x}_j$ as a row vector and $\mathbf{b}$ as a column vector.) While this eliminates the great majority of imaginable likelihood functions, it still allows estimation of many likelihood functions of interest to researchers. (The `lf` method is actually slightly less restrictive in that it allows you to define multiple linear forms; we will discuss this at greater length later in this article.)

This restriction has two benefits: (1) it allows optimization to proceed more quickly in terms of computer time, and (2) it makes the program you write to evaluate $\ln l$ simpler.

The reason for the performance benefit is that numerical optimizers (or, at least the class of numerical optimizers used by Stata) require derivatives $\partial \ln l/\partial \mathbf{b}$ and $\partial^2 \ln l/\partial \mathbf{b}^2$. The first is a k-dimensional vector and the second a $k \times k$ symmetric matrix. Calculating the numerical derivatives in the general problem requires $k + k(k+1)/2$ calculations. With the linear-form restriction, however, one can write

$$\frac{\partial \ln l}{\partial \mathbf{b}} = \frac{\partial \ln l}{\partial(\mathbf{x}_j\mathbf{b})}\,\mathbf{x}_j$$

$$\frac{\partial^2 \ln l}{\partial \mathbf{b}^2} = \frac{\partial^2 \ln l}{\partial(\mathbf{x}_j\mathbf{b})^2}\,\mathbf{x}_j'\mathbf{x}_j$$

Thus, `ml` need only calculate two numerical derivatives per iteration. Even for moderate values of k the time savings is substantial. Consider a model with 2 RHS variables and an intercept, meaning $k = 3$. Without this restriction, an optimizer must calculate 9 numerical derivatives instead of just 2, meaning 4.5 times the work. For a model with $k = 12$, the general case requires 90 derivative calculations or 45 times the 2 required with the linear-form restriction.

The linear-form restriction makes the programming of the likelihood function easier because the program you write need only receive $\mathbf{x}_j\mathbf{b}$ and not $\mathbf{x}_j$ and $\mathbf{b}$ separately. The linear form $\mathbf{x}_j\mathbf{b}$ is nothing more than a scalar value per observation and so can be stored as a variable in the dataset. Thus, your log-likelihood evaluation program need not concern itself with the identities of the variables in the model or even how many variables there are. A typical log-likelihood evaluation program reads, in its entirety,

```
program define myll
        local lnf "`1´"
        local Xb  "`2´"
        quietly replace `lnf´ = some function of $S_mldepn and `Xb´
end
```

The program receives two arguments: `` `1´ ``, the name of the variable where $\ln l(\mathbf{x}_j\mathbf{b};y_j)$ is to be stored, and `` `2´ ``, the name of a variable containing $\mathbf{x}_j\mathbf{b}$. The name of the variable corresponding to y_j (if there is such a variable) is passed in the global macro `S_mldepn`. The first two lines of our sample program simply assign more readable names to the two arguments and then the third line calculates the likelihood values. This program could be made even shorter:

```
program define myll
        quietly replace `1´ = some function of $S_mldepn and `2´
end
```

although it is not quite as readable.

Case 1: Single equations

Let us consider likelihoods that meet the linear-form restriction $\ln l(\mathbf{b}; y_j, \mathbf{x_j}) = \ln l(\mathbf{x}_j\mathbf{b}; y_j)$. Binary probit and logit fall into this category. You might be tempted to think that linear regression also falls into this category because

$$y_j = \mathbf{x}_j\mathbf{b} + u_j$$

so the log-likelihood for an individual observation is

$$\ln l(\mathbf{b}; y_j, \mathbf{x}_j) = \ln f(y_j - \mathbf{x}_j\mathbf{b})$$

where f is the (unstandardized) normal density. This likelihood was obtained by rewriting the regression as $u_j = y_j - \mathbf{x}_j\mathbf{b}$ and then noting that u_j is distributed normally.

Linear regression, however, is not an example of a `lf` single equation because, in addition to the parameters $\mathbf{b}$, the variance σ^2 of the normal density is also to be estimated. We will get to that case later.

For purposes of describing the programming of maximum-likelihood estimators let us define

> **Single Equation:** There is a one-to-one correspondence between the parameters $\mathbf{b}$ to be estimated and the RHS variables $\mathbf{x}_j$, and the way $\mathbf{x}_j$ enters the model is via the inner product $\mathbf{x}_j\mathbf{b}$. In addition, there is some outcome variable y_j (with no estimated parameters associated with it).

Linear regression is thus not "single equation" because in addition to $\mathbf{b}$, σ^2 is also to be estimated and it has no corresponding $\mathbf{x}$ variable. Linear regression when σ^2 is known, however, would be an example of a "single equation".

Here is the outline for programming single-equation estimators. First, define the program to calculate the log-likelihoods:

```
program define myll
        local lnf "`1´"
        local Xb  "`2´"
        quietly replace `lnf´ = some function of $S_mldepn and `Xb´
end
```

where some function is the log-likelihood for an individual observation. The program `myll` receives two arguments: the first is the name of a variable into which `myll` is to store the calculated log-likelihood and the second is the name of a variable into which `ml` has already stored the values of $\mathbf{x}_j\mathbf{b}$ (using the current estimate of $\mathbf{b}$). The name of the first variable is put into the local macro `lnf`, and the name of the second variable is put into the local macro `Xb`; again, this is only done to improve the readability of the code. The global macro `S_mldepn` contains the name of the outcome (dependent) variable.

Second, one estimates the model by typing

```
. ml begin
. ml function myll
. ml method lf
. eq myeq: lhsvar rhsvar1 rhsvar2 ...
. ml model b = myeq             (b is a name of your choosing; I like b)
. ml sample mysamp              (mysamp could be any new variable name)
. ml maximize f V               (f and V are names of your choosing; I like the names f and V)
. ml post myest                 (myest could be any name of your choosing)
. ml mlout myest
```

Let me explain the purpose of each of these commands:

`ml begin` merely says "Here begins a new `ml` problem; forget any previous `ml` problem I have told you."

`ml function myll` declares that the program we have written to calculate the log-likelihood is called `myll`. We do not have to call the program `myll`, but whatever we have named the program, we have to tell `ml`.

`ml method lf` is to be coded just like that. This statement says to use the `lf` method for maximizing the likelihood function.

`eq myeq:` *lhsvar rhsvar1 rhsvar2* ... states the "equation" to be estimated. `eq` is not really a `ml` command; `eq` is just a command of Stata that stores "equations"; it is the next statement that links the equation to `ml`.

`ml model b = myeq` links the equation to what `ml` is supposed to do. When we gave the `eq` command, we defined an equation named `myeq` that contained "*lhsvar rhsvar1 rhsvar2* ...", and when we gave the `ml model b = myeq` command, we told `ml` that it is to use the equation named `myeq`. For instance, we might type

```
. eq myeq: foreign mpg weight
. ml model b = myeq
```

The result of this is to define foreign as the outcome variable and mpg and weight as the covariates with associated parameters. In math, we have just stated

$$\text{log-likelihood} = \ln l(b_0 + b_1\texttt{mpg}_j + b_2\texttt{weight}_j; \texttt{foreign}_j)$$

where b_0, b_1, and b_2 are to be estimated. The definition of $\ln l$ is given by our `myll` program and `ml` knows that because we previously declared `ml function myll`.

The name `b` in `ml model b = myeq` is any name of our choosing; it is the name under which `ml` will store the estimated parameter vector, and I typically name the vector `b`. What you name the vector makes no difference. If the vector specified already exists, it will be replaced. If it does not exist, it will be created.

`ml sample mysamp` tells `ml` to identify the estimation sample now. `mysamp` is the name of a new variable that `ml` will create, and its only purpose is so that later `ml` will know which observations to use and which to ignore. In this case, all `ml sample mysamp` does is look for missing values of foreign, mpg, and weight, but if we wanted to restrict the sample in some other way, now is when we would do that:

```
. ml sample mysamp if rep78>=3
```

`ml maximize f V` performs the optimization, which is to say, finds `b` to maximize the likelihood function. To do this, `ml` needs two more names to work with and I supplied `f` and `V`. `f` will be the name of a scalar containing the optimized value of the log-likelihood. `V` will be a matrix containing the negative inverse of second derivatives (the estimated variance matrix). I could name `f` and `V` anything but typically choose `f` and `V`.

After `ml maximize f V` is executed, the estimates exist.

Vector `b` (named `b` because of the `ml model` statement) contains the estimated parameters.

Scalar `f` (named `f` because of the `ml maximize` statement) contains the log-likelihood value.

Matrix `V` (named `V` because of the `ml maximize` statement) contains the variance matrix.

I could just look at each of these using Stata's `scalar list` and `matrix list` commands. Thus, the commands that follow the `ml maximize` are optional; they simply make the output more readable.

`ml post myest` tells `ml` to save these estimation results as "official Stata estimation results" and call them `myest`. I could call them anything I wanted. For interactive use, the name plays little role so I just use `myest`. `ml post` is one step along the way to displaying pretty output, but it does something else, too. By posting the results as "official", I can now use any post-estimation command on them. I could, for instance, later use `test`.

`ml mlout myest` displays the estimation results.

The `ml` commands themselves are straightforward although you must remember to give them in the right order. Some variation is allowed but the rules are so complicated I just give them in the order shown — but sometimes I will move the `eq` command to the top:

```
. eq myeq: lhsvar rhsvar1 rhsvar2 ...
. ml begin
. ml function myll
. ml method lf
. ml model b = myeq
. ml sample mysamp
. ml maximize f V
. ml post myest
. ml mlout myest
```

Case 2: Two equations

Let us now consider a variation on the single-equation model:

$$\text{log-likelihood} = \ln l(\mathbf{x}_{1j}\mathbf{b}_1, \mathbf{x}_{2j}\mathbf{b}_2; y_{1j}, y_{2j})$$

Again I am defining "equation" in the same specific way I did in Case 1; there is a one-to-one correspondence between the estimated parameters (now $\mathbf{b}_1$ and $\mathbf{b}_2$) and the RHS variables (now $\mathbf{x}_1$ and $\mathbf{x}_2$). There need not be the same number of variables in $\mathbf{x}_1$ and $\mathbf{x}_2$.

The program to calculate the likelihoods now reads

```
program define myll
        local lnf  "`1´"
        local X1b1 "`2´"
        local X2b2 "`3´"

        local y1 : word 1 of $S_mldepn
        local y2 : word 2 of $S_mldepn

        quietly replace `lnf´ = some function of `y1´, `y2´, `X1b1´, `X2b2´
end
```

As compared to the single-equation case, we are now passed three arguments (where to store the log-likelihood calculation along with $\mathbf{x}_1\mathbf{b}_1$ and $\mathbf{x}_2\mathbf{b}_2$) and `$S_mldepn` now contains two variable names. The latter is rather inconvenient because we have to take it apart. To separate the variable names in `$S_mldepn`, we use the extended macro function `word` *#* `of` *string*. This inconvenience is overshadowed by the fact that the code above easily can be generalized to any number of equations.

In terms of what we type to estimate the model, that now becomes

```
. ml begin
. ml function myll
. ml method lf
. eq myeq1: lhsvar1 rhsvar11 rhsvar12 ...
. eq myeq2: lhsvar2 rhsvar21 rhsvar22 ...
. ml model b = myeq1 myeq2
. ml sample mysamp
. ml maximize f V
. ml post myest
. ml mlout myest
```

This is nearly identical to what we typed in the single-equation case, the differences being

1. where we previously defined only one equation, now we define two;
2. where we previously typed `ml model b = myeq`, now we type `ml model b = myeq1 myeq2` because there are two equations.

Case 3: Three and more equations

The two-equation case generalizes to any number of equations. If you have three equations, your `myll` program will receive four arguments (where to store the result along with the evaluations of $\mathbf{x}_1\mathbf{b}_1$, $\mathbf{x}_2\mathbf{b}_2$, and $\mathbf{x}_3\mathbf{b}_3$); `$S_mldepn` will contain three variable names (y_1, y_2, and y_3); and in terms of the `ml` commands, you just define all three equations and then, when you define the model, you type

```
. ml model b = myeq1 myeq2 myeq3
```

Case 4: Ancillary parameters

So far we have restricted ourselves to cases where there is a one-to-one correspondence between estimated parameters $\mathbf{b}$ and explanatory variables $\mathbf{x}$. As I mentioned earlier, often there are more parameters than that. In linear regression, the log-likelihood is

$$\text{log-likelihood} = -\frac{1}{2}\Big(\frac{y_j - \mathbf{x}_j\mathbf{b}}{\sigma}\Big)^2 - \ln(\sqrt{2\pi}\,\sigma)$$

where, in addition to $\mathbf{b}$, there is an extra parameter σ to be estimated.

I am going to show you the easiest way to deal with these ancillary parameters, but before I do, I want to show you that you could estimate this models with nothing more than what you already know. I want to show you this not as an aside but because it is important to understanding the way we actually do it.

The trick is to introduce a second equation and then mostly to ignore it. The second equation would be

$$\text{LHS} : \text{trash}_j; \quad \text{RHS} : \sigma\mathbf{x}_{2j}, \quad \text{where } \mathbf{x}_{2j} = 1 \text{ for all } j$$

That is, pretend we tricked `ml` into thinking there are two equations. That does not mean there mathematically have to be two equations; since we get to write the program `myll` that calculates the log-likelihood, what we do with the second equation is up to us. Let's assume the likelihood we are really interested in maximizing is

$$\text{log-likelihood} = \ln l(\mathbf{x}_j\mathbf{b}, \sigma; y_j)$$

To make it fit into two-equation framework, we rewrite this as

$$\text{log-likelihood} = \ln l(\mathbf{x}_j\mathbf{b}, \mathbf{x}_{2j}\sigma; y_j, \text{trash}_j)$$

where $\mathbf{x}_{2j} = 1$ for all j and the contents of trash_j are irrelevant; both of these variables are merely introduced to satisfy `ml`'s constraints.

Here are the `ml` commands we would use:

```
. gen trash = (pick a number, say 42)
. gen one = 1
. ml begin
. ml function myll
. ml method lf
. eq myeq: lhsvar rhsvar1 rhsvar2 ...
. eq myeq2: trash one              (the phony equation)
. ml model b = myeq myeq2
. ml sample mysamp
. ml maximize f V
. ml post myest
. ml mlout myest
```

Here is how we would write `myll`:

```
program define myll
        local lnf   "`1´"
        local Xb    "`2´"
        local sigma "`3´"
        local y : word 1 of $S_mldepn
        quietly replace `lnf´ = some function of `y´, `Xb´, `sigma´
end
```

That is, in `myll`, we would simply ignore the second dependent variable **`trash`**. We have arranged for the second equation to equal σ by making it $\texttt{one}_j \times \sigma = 1 \times \sigma = \sigma$.

What I have written above will not work, but that is only because I ignored an implicit assumption of my previous definition of an "equation". I said that an "equation" has a one-to-one correspondence between parameters $\mathbf{b}$ to be estimated and the RHS variables $\mathbf{x}$. This assumes that $\mathbf{x}$ includes a constant $x_{0j} = 1$, because, unless you take some special action, `ml` automatically includes an intercept in each of your equations.

That is, when we typed

```
. eq myeq: foreign mpg weight
```

we defined

1. `foreign` as the outcome variable;
2. `mpg` and `weight` as the covariates;
3. the linear combination $b_0 + b_1\texttt{mpg} + b_2\texttt{weight}$ as the evaluation of the “equation”.

There is a way to define an “equation” with no intercept, but you seldom want that. Most “equations” include an intercept.

You can also define equations to include only an intercept. Consider the statement

```
. eq myeq: foreign
```

which means

1. `foreign` is the outcome variable;
2. there are no covariates;
3. the linear combination b_0 is the evaluation of the “equation”.

So here is how we could estimate the log-likelihood $\ln l(\mathbf{x}_j\mathbf{b}, \sigma; y_j)$:

```
. gen trash = (pick a number, say 42)
. ml begin
. ml function myll
. ml method lf
. eq myeq: lhsvar rhsvar1 rhsvar2 ...
. eq myeq2: trash                     (the phony equation)
. ml model b = myeq myeq2
. ml sample mysamp
. ml maximize f V
. ml post myest
. ml mlout myest
```

and our corresponding likelihood-evaluation program would read

```
program define myll
        local lnf   "`1'"
        local Xb    "`2'"
        local sigma "`3'"
        local y : word 1 of $S_mldepn
        quietly replace `lnf' = some function of `y', `Xb', `sigma'
end
```

To summarize, an ancillary parameter can be handled by introducing an equation for it. The newly introduced equation contains only an intercept. To make this work, we had to introduce new variable trash containing literally anything. We needed it as a placeholder in the `eq` statement, but thereafter we ignored it.

Actually, we do not have to provide a trash variable because `ml` can be told that there is no dependent variable associated with an equation. Hence, here is a better way we could code this problem (I’ve marked the statements that have changed):

```
                                        (we no longer generate trash)
. ml begin
. ml function myll
. ml method lf
. eq myeq: lhsvar rhsvar1 rhsvar2 ...
. eq sigma:                             (changed)
. ml model b = myeq sigma, depv(10)     (changed)
. ml sample mysamp
. ml maximize f V
. ml post myest
. ml mlout myest
```

Note the `depv(10)` option on the `ml model` statement. The 10 does not mean the number ten, it means 1 and 0. It states that the first equation (`myeq`) has 1 dependent variable and the second equation (`sigma`) has none.

Our `myll` program can now read

```
program define myll
        local lnf   "`1´"
        local Xb    "`2´"
        local sigma "`3´"
        quietly replace `lnf´ = some function of $S_mldepn, `Xb´, `sigma´
end
```

If you compare this with the previous example, you will see that we have omitted

```
        local y : word 1 of $S_mldepn
```

`$S_mldepn` now contains only one variable name so we can just use it directly. The previous version of `myll` would also work; it is just that decomposing `$S_mldepn` is no longer necessary.

To summarize, each ancillary parameter is handled by introducing a new "equation"—an "equation" with no dependent variable—into the code. This is not just trickery; there is a real advantage to thinking this way. We have just written a program to maximize

$$\text{log-likelihood} = \ln l(\mathbf{x}_j\mathbf{b}, \sigma; y_j)$$

Say that later we wish to consider a variation on this model and let $\sigma_j = \mathbf{x}_{2j}\mathbf{b}_2$. That is, we want to parameterize σ and let it vary with $\mathbf{x}_2$. We can do this and we do not have to rewrite `myll`! All we do is change what we type when we define the equations; instead of typing

```
. eq myeq: lhsvar rhsvar1 rhsvar2 ...
. eq sigma:
. ml model b = myeq sigma, depv(10)
```

we could type

```
. eq myeq: lhsvar rhsvar1 rhsvar2 ...
. eq sigma: x2var1 x2var2 ...
. ml model b = myeq sigma, depv(10)
```

Case 5: Multiple ancillary parameters

Multiple ancillary parameters are handled by simply introducing more equations. Consider a problem with two ancillary parameters:

$$\text{log-likelihood} = \ln l(\mathbf{x}_j\mathbf{b}, \sigma_1, \sigma_2; y_j)$$

where $\mathbf{b}$, σ_1, and σ_2 are to be estimated. The estimation code reads

```
. ml begin
. ml function myll
. ml method lf
. eq myeq: lhsvar rhsvar1 rhsvar2 ...
. eq sigma1:
. eq sigma2:
. ml model b = myeq sigma1 sigma2, depv(100)
. ml sample mysamp
. ml maximize f V
. ml post myest
. ml mlout myest
```

and the corresponding `myll` program reads

```
program define myll
        local lnf    "`1´"
        local Xb     "`2´"
        local sigma1 "`3´"
        local sigma2 "`4´"
        quietly replace `lnf´ = some function of $S_mldepn, `Xb´, `sigma1´, `sigma2´
end
```

Case 6: Multiple ancillary parameters and multiple equations

This can all be put together. Consider

$$\text{log-likelihood} = \ln l(\mathbf{x}_{1j}\mathbf{b}, \mathbf{x}_{2j}\mathbf{b}, \sigma_1, \sigma_2; y_{1j}, y_{2j})$$

where $\mathbf{b}_1$, $\mathbf{b}_2$, σ_1, and σ_2 are to be estimated. Here we have two real equations to be estimated and two ancillary parameters. The estimation code reads

```
. ml begin
. ml function myll
. ml method lf
. eq myeq1: lhsvar1 rhsvar11 rhsvar12 ...
. eq myeq2: lhsvar2 rhsvar21 rhsvar22 ...
. eq sigma1:
. eq sigma2:
. ml model b = myeq1 myeq2 sigma1 sigma2, depv(1100)
. ml sample mysamp
. ml maximize f V
. ml post myest
. ml mlout myest
```

and the corresponding `myll` program reads

```
program define myll
        local lnf    "`1´"
        local X1b1   "`2´"
        local X2b2   "`3´"
        local sigma1 "`4´"
        local sigma2 "`5´"
        local y1 : word 1 of $S_mldepn
        local y2 : word 2 of $S_mldepn
        quietly replace `lnf´ = some function of `y1´, `y2´, `X1b1´, `X2b2´, `sigma1´, `sigma2´
end
```

The constant() option: Specifying whether equations include an intercept

We have discussed creation of equations containing an intercept only and no dependent variable; that is how we handle ancillary parameters. Let us now consider equations with a dependent variable but having no intercept. This has nothing to do with ancillary parameters; sometimes we simply want to estimate models without an intercept (`nocons` in Stata parlance).

In particular, let us consider

$$\text{log-likelihood} = \ln l(\mathbf{x}_j\mathbf{b}, \sigma; y_j)$$

where $\mathbf{b}$ and σ are to be estimated, and $\mathbf{b}$ contains no intercept. We write the program to evaluate the log-likelihood in the same way as if $\mathbf{b}$ did contain an intercept:

```
program define myll
        local lnf   "`1´"
        local Xb    "`2´"
        local sigma "`3´"
        quietly replace `lnf´ = some function of $S_mldepn, `Xb´, `sigma´
end
```

The specification that $\mathbf{b}$ contains no intercept affects only the `ml model` statement:

```
. ml begin
. ml function myll
. ml method lf
. eq myeq: lhsvar rhsvar1 rhsvar2 ...
. eq sigma:
. ml model b = myeq sigma, depv(10) constant(01)          (changed)
. ml sample mysamp
. ml maximize f V
. ml post myest
. ml mlout myest
```

Note the `constant(01)` option, which is read as constant 0 and 1. The first digit refers to the first equation and the second to the second. The constant 0 for the first equation means that there is to be no intercept in `myeq`. The constant 1 for the second equation means that there is to be an intercept in `sigma`.

The rules for filling in the `constant()` option are

1. It has as many arguments are there are equations. The first argument corresponds to the first equation, the second to the second equation, and so on.
2. The arguments are the digits 1 or 0, and the arguments are run together. Thus, 101 is three arguments.
3. An argument of 1 means the equation is to contain a constant. An argument of 0 means the equation is not to contain a constant.
4. If you omit the `constant()` option, results are as if you coded `constant(11...1)`; that is, all equations are to contain a constant.

The depv() option: specifying the number of dependent variables

When you specify an equation

```
. eq myeq: varname1 varname2 varname3 ...
```

varname1 is normally interpreted as being the dependent variable and the remaining names are interpreted as being independent variables. The equation could just as well be interpreted as containing no dependent variables; that is, *varname1*, *varname2*, …, could all be interpreted as being independent variables. Alternatively, the equation could be interpreted as containing two dependent variables, *varname1* and *varname2*, and the remaining variables interpreted as independent variables.

How the equation is interpreted is determined by the `depv()` option on the `ml model` statement. In dealing with ancillary parameters — when we wanted the equation named `sigma` interpreted as containing no dependent variables — we coded

```
. model b = myeq sigma, depv(10)
```

The `depv()` arguments work like the `constant()` arguments; the digits are positional. We specified `depv()` as 1 for the first equation and 0 for the second, meaning `myeq` was to be interpreted as containing one dependent variable and `sigma` interpreted as containing no dependent variables. Unlike the `constant()` option, we are not limited to the digits 0 and 1. The possible arguments for `depv()` are 0, 1, 2, …, 9.

Consider a likelihood function with two dependent variables but only one set of explanatory variables (and an ancillary parameter):

$$\text{log-likelihood} = \ln l(\mathbf{x}_j\mathbf{b}, \sigma; y_{1j}, y_{2j})$$

where $\mathbf{b}$ and σ are to be estimated. For instance, the true dependent variable y_j might be unobserved and y_{1j} and y_{2j} might be the observed bounds of an interval known to contain y_j. The program to evaluate this likelihood function would read

```
program define myll
        local lnf   "`1´"
        local Xb    "`2´"
        local sigma "`3´"
        local y1 : word 1 of $S_mldepn
        local y2 : word 2 of $S_mldepn
        quietly replace `lnf´ = some function of `y1´, `y2´, `Xb´, `sigma´
end
```

and the corresponding `ml` commands would be

```
. ml begin
. ml function myll
. ml method lf
. eq myeq: y1var y2var rhsvar1 rhsvar2 ...
. eq sigma:
. ml model b = myeq sigma, depv(20)
. ml sample mysamp
. ml maximize f V
. ml post myest
. ml mlout myest
```

The `depv(20)` option on the `ml model` statement specifies that the first equation (`myeq`) contains two dependent variables and the second equation (`sigma`) none.

The eq command

In all the examples above I have specified equations by typing

```
. eq myeq: varname1 varname2 ...
```

and then later I have referred to the equation by typing

```
. ml model b = myeq ...
```

`eq` allows a shorthand. I can omit the name of the equation and simply type

```
. eq varname1 varname2 ...
```

and doing this is equivalent to typing

```
. eq varname1: varname1 varname2 ...
```

That is, the first variable specified plays a dual role, being part of the contents of the equation and being the name of the equation as well. If I adopted this shorthand, then I would have to remember to specify the correct name of the equation when I referred to it on the `ml model` statement:

```
. ml model b = varname1 ...
```

For instance, I could type

```
. eq myeq: foreign mpg weight
. ml model b = myeq
```

or I could type

```
. eq foreign mpg weight
. ml model b = foreign
```

It makes no difference.

Calculating log-likelihood functions

In the log-likelihood programs shown above we have shown the actual calculation of the function taking place on one line. For instance, in the single-equation with ancillary parameter case, we showed,

```
program define myll
        local lnf   "`1´"
        local Xb    "`2´"
        local sigma "`3´"
        quietly replace `lnf´ = some function of $S_mldepn, `Xb´, `sigma´
end
```

In practice, real likelihood functions are difficult to write in a single `replace` statement. Obviously, one can split the calculation into multiple statements; it is merely important that, at the conclusion of your program, `` `lnf´ `` be filled in. Temporary variables are often used:

```
program define myll
        local lnf   "`1´"
        local Xb    "`2´"
        local sigma "`3´"
        tempvar part1 part2 part3
        quietly gen double `part1´ = some function of $S_mldepn, `Xb´, `sigma´
        quietly gen double `part2´ = some function of $S_mldepn, `Xb´, `sigma´
        quietly gen double `part3´ = some function of $S_mldepn, `Xb´, `sigma´
        quietly replace `lnf´ = `part1´*(`part2´+`part3´)
end
```

The important thing to note here is that all intermediate calculations are stored in double precision. The temporary variables `` `part1' ``, `` `part2' ``, and `` `part3' `` were defined by `generate double` statements, not mere `generate`s. Had the `double` been omitted, the intermediate values would have been stored in float precision. Float precision is not adequate for the numerical optimization technique used by the `lf` method.

It is also important to remember that you are to return the log of the likelihood, not the likelihood itself. For some likelihood functions, this simply amounts to taking the `ln()` of a calculated result. Whenever possible, however, it is better to evaluate the log function analytically. For example, it is a bad idea to code the log-likelihood for linear regression as

$$\text{log-likelihood} = \ln\Big(\frac{1}{\sqrt{2\pi}\,\sigma}\ \exp\Big[-\frac{1}{2}\Big(\frac{y_j - \mathbf{x}_j\mathbf{b}}{\sigma}\Big)^2\Big]\Big)$$

One should code the formula after analytic simplification:

$$\text{log-likelihood} = -\frac{1}{2}\Big(\frac{y_j - \mathbf{x}_j\mathbf{b}}{\sigma}\Big)^2 - \ln(\sqrt{2\pi}\,\sigma)$$

You must remember that the `lf` method will use your program not just to obtain values for the log-likelihood function, but that it will use the values returned by your function to calculate numeric derivatives. You want your function to evaluate correctly for as wide a range of values as you can manage and taking logs analytically typically arranges that.

People who do not have a numerical background often assume that $z = \ln(\exp(z))$ is true, at least approximately and that is the responsibility of good software to ensure the statement is adequately true. In Stata, for instance, `ln(exp(-100))` differs from -100 by less than 1.5×10^{-14}. But now consider `ln(exp(-800))`. `exp(-800)` evaluates to exactly 0 (the true answer is less than 10^{-320}) and `ln(0)` evaluates to missing. Similarly, $z =$ `ln(exp(`z`))` stops being even approximately true for $z \geq 705$ because `exp(705)` is missing (it is too large). Only thoughtful programming can avoid such situations from arising.

os16.1	Importing Stata graphs into word processors on the Macintosh: Part 2

Chinh Nguyen, Stata Corp., FAX (409) 696-4601, tech@stata.com

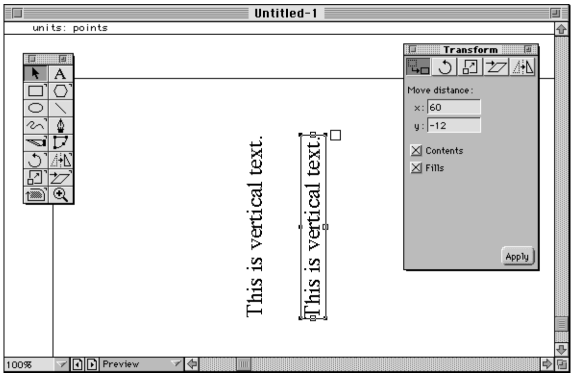
In STB-32 *os16*, I discussed how to get around the limitations of the Macintosh and Stata in getting publication-quality results with Stata's graphs. I also compared how the top DTP packages faired importing Stata's graphs. Stata 5.0 has been released since that time and many of the limitations of the Macintosh and Stata raised in the article have been addressed. Namely, Stata can now use Macintosh and Windows fonts in its graphs. Stata's method of drawing circles has been changed as well to accommodate peculiarities in Microsoft Word, Microsoft PowerPoint, and Deneba Canvas. Although these improvements make the results of importing Stata's graphs much better, there are still some limitations with Word 6.0 that users should be aware of. In this article, I will discuss those limitations, their causes, and how to avoid them.

The Macintosh and vertical text

The ability to use Macintosh and Windows fonts in Stata's graphs was one of the most requested features for Stata prior to 5.0 and was one of the first things we worked on. Implementing it was more difficult on the Macintosh because of a very surprising limitation — the Macintosh has no built-in support for handling vertical text. So how does Stata draw vertical text? It does this by drawing the text horizontally into a bitmap, rotating it by −90°, and then stamping it onto the graph — the same method used by other Macintosh applications. This method works well for displaying and printing graphs from Stata but is not ideal for copying graphs to the Clipboard. Because the vertical text is a bitmap, it cannot be directly edited as text from a DTP package. In most cases, the inability to edit the vertical text is not that great of a limitation since Stata usually puts it in one area. You can simply replace the text by removing the bitmap from the graph and inserting your own vertical text in its place using a DTP package.

Another shortcoming this method of drawing vertical text is that it will only print it at the resolution of the bitmap rather than the resolution of the printer. If you have a 1200 dpi (dots per inch) printer but the bitmap was drawn at a resolution of 300 dpi, the best it would be printed at is 300 dpi. This only applies to the vertical text of Stata graphs printed from another application — the rest of the graph prints at the highest resolution possible. Graphs printed from Stata always print at the highest resolution possible including vertical text. For drawing vertical text into the Clipboard, we chose 300 dpi for the resolution as a compromise between print quality and memory limitations. Anything much higher would potentially cause memory crashes.

You may be thinking to yourself that the Macintosh does support vertical text because you can easily add it to your documents using your favorite DTP package. That's not entirely true. It is internally supported by your DTP package but outside of your DTP package, it is not recognized as true vertical text. If you were to copy vertical text from one package and paste it into another, you would either get horizontal text, horizontal text stacked vertically, or a bitmap representation of the text. Figure 1 shows a FreeHand 5.5 document with vertical text that has been selected, copied, and pasted back into the document. Figure 2 shows the same text after it was copied from FreeHand and pasted into PageMaker 6.0. You'll notice that the text is now horizontal. This is the reason that Stata chose to draw the text as a bitmap rather than draw it as horizontal text. If the text were drawn horizontally, it would require that every imported graph would have to be edited and the text on the y-axis rotated.



Move distance:
x: 60
y: -12
☒ Contents
☒ Fills
Apply

Figure 1. FreeHand 5.5 document.

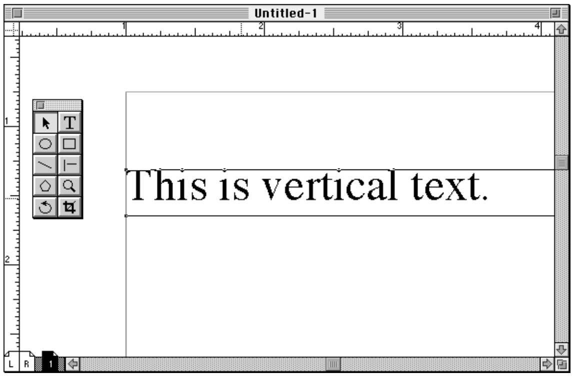


Figure 2. PageMaker 6.0 document.

There was the possibility of drawing the text horizontally and stacking it vertically but there were a few problems with this method. The idea behind this method was to have each letter of text drawn on a separate line. The problem was that when the text was drawn as one string (so that it may be edited as a whole later), some letters such as 'i' and 'l' were so narrow that they would appear on the same line. That made the text very difficult to read and visually unappealing. There was always the possibility of drawing the text one character at a time but that would have made each character an individual drawing object. If it was edited later, you would have to select each character object for modification rather than one single object (which is what I believe happens with Excel). Besides the obvious problems with this method, a graph in Stata for the Macintosh would look different from a graph in Stata for Windows. This would introduce an inconsistency across Statas on different platforms — something that greatly concerns us and a situation we try to stay away from as much as possible.

Word 6.0 and Stata graphs

Users of Word 6.0 for the Macintosh have another problem when importing Stata graphs. Pasting a graph into Word document and then printing it *basically* gives the intended results. But Word introduces a few oddities when it imports the graphs. As an example, Figure 3 illustrates a Stata graph as it is directly printed from Stata. We would expect that Word would give us the exact same output but Figure 4 shows what happens after a graph has been pasted to and printed from Word.

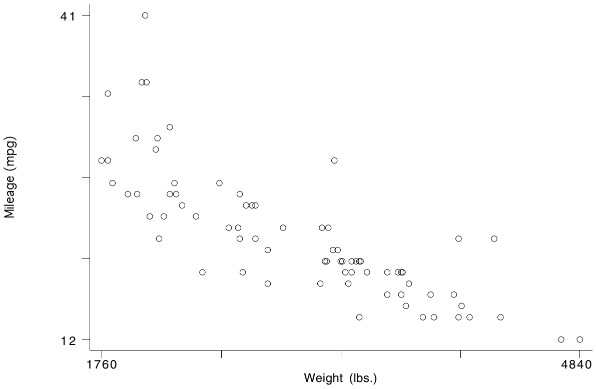


Figure 3. Graph printed from Stata.

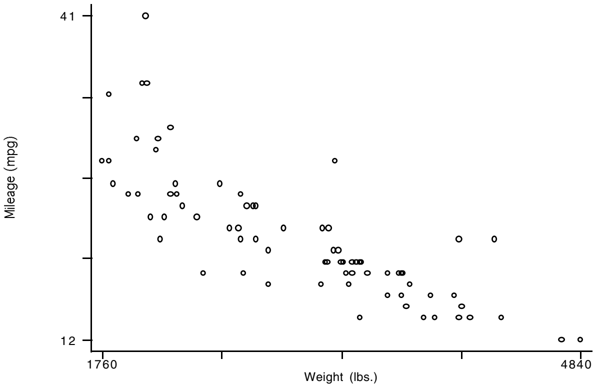


Figure 4. Graph printed from Word.

The first difference you'll notice is that Word prints the graphs with thicker lines. That's not such a problem as we would have increased the line thickness anyway. But if you would have wanted thinner lines, it wouldn't be possible with Word without tedious editing. The most obvious problem with Word's output is that some of the symbols in the scatterplot printed from Word are ovals rather than circles. What is the cause of this? Perhaps the method for drawing the graph for printing is different from the method for drawing the graph for the Clipboard? No. The exact same routines for drawing the graph are used regardless of whether it is to be printed, drawn to the screen, or drawn to the Clipboard. The only differences are the characteristics of the destination to be drawn to.

I initially discovered this problem while testing the graph copying feature of Stata with Word 6.0. To verify the fault was not with Stata, I switched to the Clipboard viewer from the desktop by selecting **Show Clipboard** from the **Edit** menu. Yet, the Clipboard viewer showed the same odd problem — some of the circles appeared as ovals. Undaunted, I fired up FreeHand 5.5 and pasted the graph there. Again the circles appeared as ovals! But then I used the magnifier and zoomed in on the ovals and found they were circles. Printing gave the expected results — the circles were truly circles.

When the Macintosh shows the graph in the Clipboard viewer, it has to scale the image down from 300 dpi to 72 dpi. It appears that this scaling is causing distortions to some of the circles due to rounding error. When I changed the code to draw to the Clipboard at 72 dpi, the circles always showed up as circles in both the Clipboard and Word. So it seems that Word has an internal resolution of 72 dpi and images transferred at a higher resolution have a possibility for distortion. Other applications use a much higher resolution so you won't see the distortion you see in Word.

So, on the one hand, graphs containing vertical text copied to the Clipboard at 72 dpi print very poorly because they contain low-resolution bitmaps. On the other hand, graphs containing circles copied to the Clipboard at 300 dpi have some of their circles distorted by Word and do not show up as intended. A future solution to the conflict may be to offer a choice between 72 dpi and 300 dpi for copying to the Clipboard.

Now let's move on to the next oddity of Word. Figure 5 shows a different graph printed from Stata. Figure 6 shows the graph after it was copied to the Clipboard, pasted into a Word document and then double-clicked so it could be edited from Word's graphics editor. You'll notice a few problems. The tick marks are no longer completely vertical (marks 1.0, 4.0, and 9.0 on the x-axis) or horizontal (mark 50 on the y-axis). You'll also notice that the segments in the spline are somewhat disjointed.

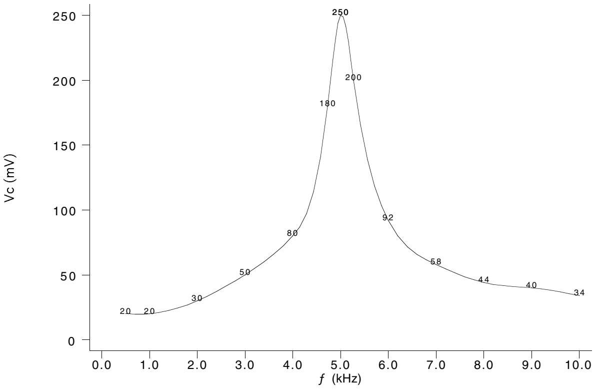


Figure 5. Graph printed from Stata.

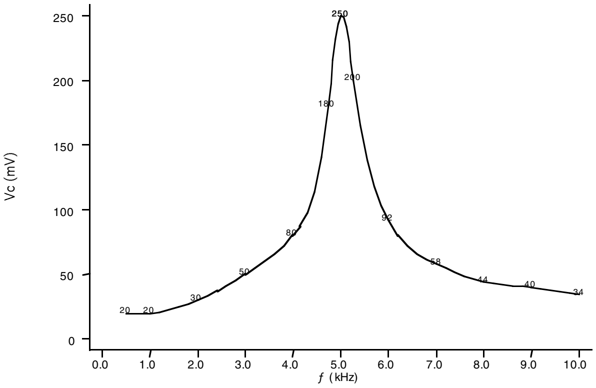


Figure 6. Graph from Word's graphic editor.

Again, this appears to be a problem with Word's internal resolution for images. When the graph is entered into Word's graphic editor, it is converted to Word's internal format for images. Word is converting the graph into its own metric and in the process, distorting many of the drawing objects due to rounding error. This situation is not unique to Stata and has been reproduced in many other packages including the highly-regarded graphing package Deltagraph 4.0 when used with Word.

Tips for better output

How can you avoid this if you use Word 6.0 and have no other drawing package available to you? You should minimize the modifications needed to your graphs by using labels and the graph options available when drawing the graphs. For example, for the graph in Figure 5, I wanted to include the symbol for frequency, f, in the x-axis title. Prior to 5.0, this was not possible because Stata only recognized standard ASCII characters. I would have had to edit the graph in another package and enter any non-ASCII characters myself. But Stata 5.0 is now 8-bit clean which allows non-ASCII characters to be entered. To create the title for the x-axis, I used Stata 5.0 and labeled the variable with the symbol f.

But there is currently nothing you can do when drawing a graph in Stata to avoid Word's distortion of the circles. You could try specifying a different plotting symbol but the same problem with Word would probably happen with other symbols. In that case, you should still go with the recommendation of the last article — create an EPS (encapsulated postscript) file of your graph. (see [U] **33 More on Stata for Macintosh**). Although it is much less convenient than simply copying and pasting, it gives the exact same output of printing the graph directly from Stata. Figure 7 shows an example of creating an EPS file in Stata and printing it from Word. You can see that it is identical to the graph in Figure 3.

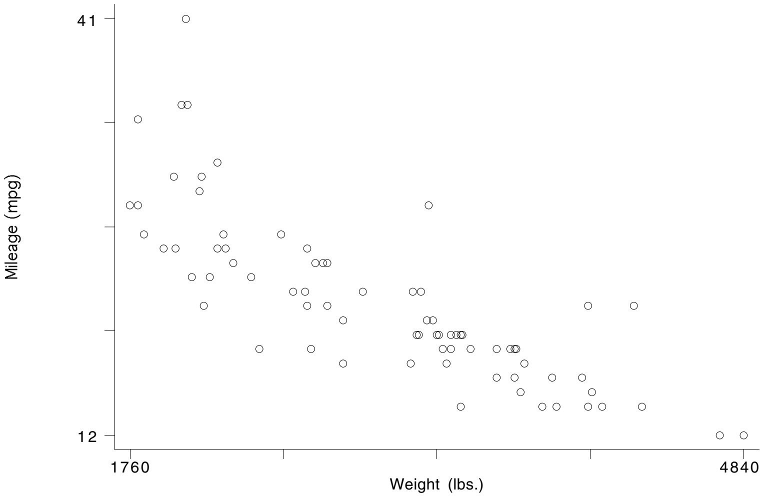


Figure 7. Graph saved as an EPS file printed from Word.

Although the output of EPS files is superior, it still has some weaknesses. It is difficult to modify the files because there are still very few DTP applications that can edit them, and EPS files tend to be much larger than the same image as a PICT file.

People complain Word 6.0 is bloated and slow, but many are forced to use it because of its popularity in business and education. Despite its faults, Word is a very capable word processor and its mishandling of hi-resolution images won't prevent you from publishing high-quality graphs if you follow some of the suggestions from this article.

sbe13 | Age-specific reference intervals ("normal ranges")

Eileen Wright, Royal Postgraduate Medical School, UK, ewright@rpms.ac.uk
Patrick Royston, Royal Postgraduate Medical School, UK, proyston@rpms.ac.uk

Introduction

Age-specific reference intervals (RIs) are in common use in medical practice, where it is assumed that values which lie outside the RI may indicate individuals with some disorder. In the analysis of aspects of body size such as height and weight, age-specific RIs are often referred to as "growth charts". Other applications include the assessment of fetal size for possible intrauterine growth retardation (Altman and Chitty 1993) and in risk screening for heart disease according to serum cholesterol concentration (Mann et al. 1988). To construct RIs, a sample from an appropriate reference population, usually assumed to consist of "normal" individuals, is needed. The measurement of interest may depend on factors such as age, sex, height, or weight, but correction for age is usually the most important. A $100p\%$ age-specific RI is defined to be the region, symmetric about the median curve in the probability scale, encompassed by the $100 \times (1-p)/2$-th and $100 \times (1+p)/2$-th centile curves. For example, a 94% RI is bounded by 3rd and 97th centile curves.

As an example, Figure 1 shows observations of fetal kidney volume, measured by ultrasonography, together with an estimated median curve and a 94% gestational age-specific RI. A description of how it was created is given later in this article.

Many methods have been proposed for the construction of age-specific RIs (see Wright and Royston (1995) for a review of some of these). The transformation of a measurement towards normality is a popular approach due to the familiarity of the normal distribution and its convenient mathematical properties. The method described here is based upon this idea. The parameters of a chosen statistical distribution, based on transformation, are modeled as fractional polynomial functions of age, and estimation is by maximum likelihood. Smooth centile curves with explicit formulae and individual SD-scores (residuals, which should have a standard normal distribution) are easily obtained.

The method is implemented as an ado-file, `xriml`, which is a regression-like command with the following basic syntax:

```
xriml yvar [xvar] [if exp] [in range], dist(distribution_code) [major_options minor_options]
```

Full details are given in the section entitled *Syntax of the xriml command.*

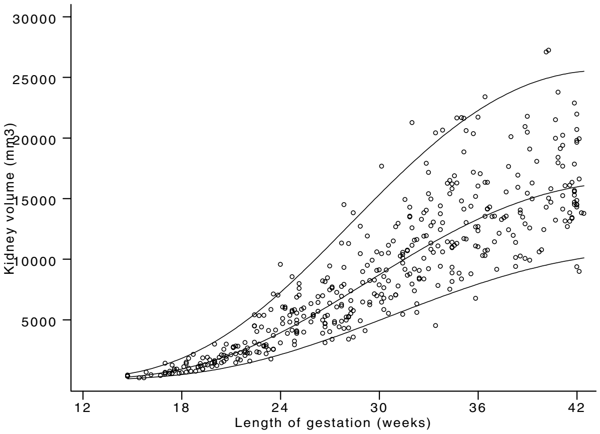


Figure 1. Kidney volume vs. age with 94% RI and median.

Method

Suppose the data consist of N pairs of observations, assumed to be a random sample from the population of interest, where Y is the measurement of interest and T is age. Y is a continuous variable and T may be continuous or ordinal. The data should be cross-sectional; i.e. there should be no more than one measurement per individual. The fetal kidney volume measurements, used to estimate the RI, are plotted against gestational age in Figure 1.

Normal model

We consider first a normal model. Let μ_T and σ_T be the age-varying population mean and standard deviation (SD) of Y as functions of T. Then the standardized variable

$$Z = \frac{Y - \mu_T}{\sigma_T}$$

has mean 0 and SD 1 at all ages. If Y is normally distributed then Z has a standard normal distribution, $N(0,1)$. A $100p$th centile curve for Y is calculated from q_p, the $100p$th percentile of $N(0,1)$ as

$$C_p = \mu_T + q_p \sigma_T$$

When the data are not normally distributed and Y has positive values, log transformation of Y will reduce positive skewness and may also reduce heteroscedasticity (age-related changes in variance). Figure 2 shows the substantial reduction in heteroscedasticity which results when fetal kidney volume is log transformed. Centile estimates calculated in the log scale should be back-transformed by exponentiation to give estimates in the original scale.

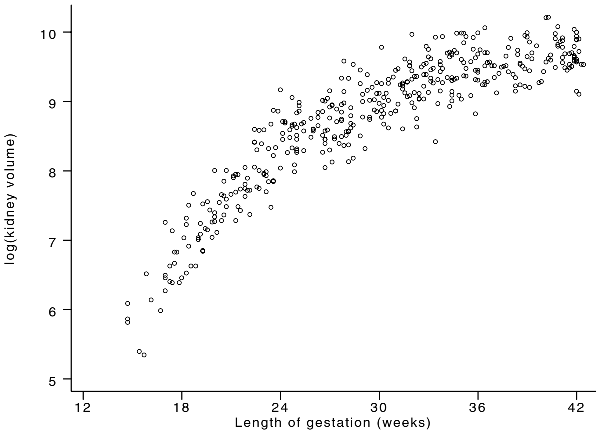


Figure 2. Log(kidney volume) vs. age.

Now consider the choice of suitable regression functions for μ_T and σ_T, which we call the M and S curves respectively. Each parameter curve is modeled using `fracpoly` (see [R] **fracpoly** in the Stata Reference Manuals), which determines the best-fitting fractional polynomial (FP) powers of T for a given *yvar*. The selection of FPs is discussed in the manual and Royston and Altman (1994).

Firstly a suitable FP is found for the mean, then for the SD, since incorrect modeling of μ_T will give inaccurate estimates of σ_T. The M curve may be found by fractional polynomial regression of Y on T. For the log fetal kidney volume data, a second degree FP with powers $(0, 0.5)$ is selected by `fracpoly`, and appears to fit well. Figure 3 shows the fitted mean curve, whose equation is $\hat{\beta}_{M;0} + \hat{\beta}_{M;1} \log T + \hat{\beta}_{M;2}\sqrt{T}$.

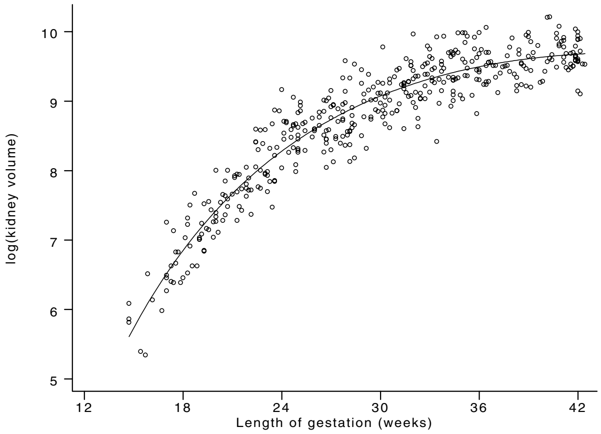


Figure 3. Mean curve with powers (0,0.5).

If Y is normally distributed, the expected value of the "scaled absolute residuals", $A = \sqrt{\pi/2}\,|Y - \mu_T|$, is equal to the SD (Altman 1993), so the fitted values from a regression of A on T may be used to estimate the S curve. The scaled absolute residuals from the mean curve fit of Figure 3 are plotted against T in Figure 4. An FP of first degree with power 3 fits the data well, so that the equation for the S curve is $\hat{\beta}_{S;0} + \hat{\beta}_{S;1}T^3$. In fact, a straight line is almost as good a fit as the FP.

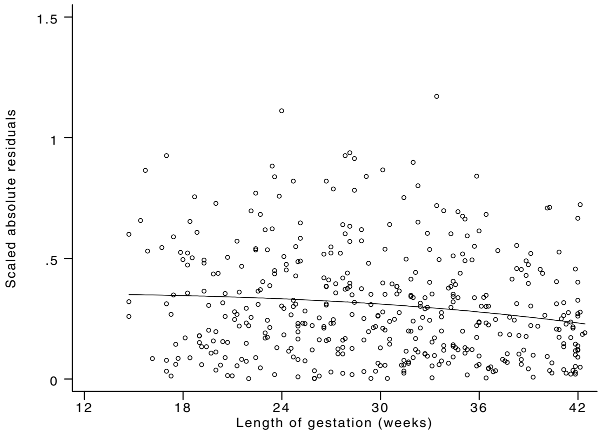


Figure 4. SD curve with power (3).

Once satisfactory powers of T have been chosen for both μ_T and σ_T, the regression coefficients for each curve may be estimated by maximum likelihood (using appropriate equations in `ml`). The best powers for each curve could be found by maximizing the likelihood using a grid search over every combination of powers, but the large number of models to be fitted makes the procedure unacceptably slow in Stata. For the 424 observations of log fetal kidney volume and gestational age, a normal model using powers $(0, 0.5)$ for M and (1) (i.e. a straight line) for S is fitted using `xriml` as follows:

```
. xriml lnkidvol gaw, fp(m:0 0.5,s:1) dist(n)

Iteration 0:  Log Likelihood = -88.611681
Iteration 1:  Log Likelihood = -88.573762
Iteration 2:  Log Likelihood = -88.573759

Normal Regression                                  Number of obs    =     424

Log Likelihood =    -88.5737590
```

```
------------------------------------------------------------------------------
lnkidvol |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
Mcurve   |
    Xm_1 |   15.28276   .8573101     17.826   0.000       13.60247    16.96306
    Xm_2 |  -4.522462   .3210681    -14.086   0.000      -5.151744    -3.89318
   _cons |  -18.13476   1.153769    -15.718   0.000      -20.39611   -15.87342
---------+--------------------------------------------------------------------
Scurve   |
    Xs_1 |  -.0043249   .0015222     -2.841   0.004      -.0073084   -.0013415
   _cons |   .4294115   .0488307      8.794   0.000       .3337051    .5251179
------------------------------------------------------------------------------

  Final deviance =   177.148 (424 observations.)
```

`xriml` silently creates new variables in the data as follows. The fitted parameter curves are stored in `M_ml` and `S_ml`, the standardized values (also known as SD-scores) are put into `Z_ml` and the default centile estimates (3rd and 97th) are placed in `C3_ml` and `C97_ml`. The routine automatically plots the centile and median curves superimposed on the raw data, as shown in Figure 5 for the log fetal kidney volume data. When it is unclear whether a higher degree FP may be a better fit for a particular parameter, it is advisable to fit both models using `xriml` and compare the difference in final deviances to a chi-squared distribution with number of degrees of freedom equal to twice the number of extra terms in the larger model. Note that in the above output, `dist(n)` indicates the fitting of a normal model—other possibilities are discussed in the section entitled *Extensions*.

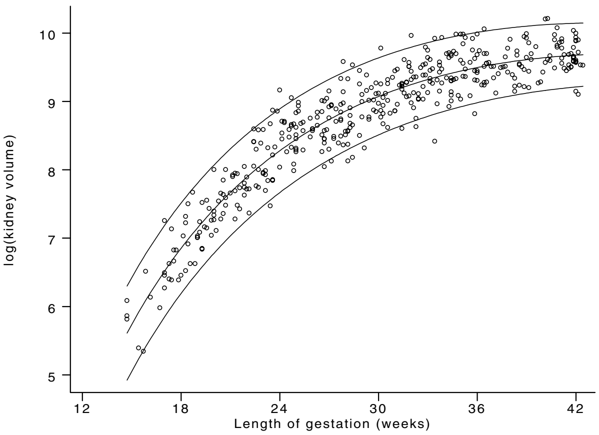


Figure 5. 94% RI and median for log(kidney volume).

It is important to assess model fit. The normality of the SD-scores in `Z_ml` may be checked subjectively using a normal plot (`qnorm`), or more formally using the Shapiro–Wilk W (`swilk`) or Shapiro–Francia W' (`sfrancia`) tests. For the log fetal kidney volume example, the W test has a p-value of 0.64. The normal plot appears reasonably linear (see Figure 6). In cases where there appear to be departures from normality, the non-normal skewness and/or kurtosis in the data need to be accounted for using a more complex model, as discussed in the next section.

To assess the fit of the M and S curves, running line smooth curves (`running`, Sasieni 1995) of `Z_ml` and `Z_ml` squared against T may, respectively, show any obvious trends or departures from the expected values of 0 and 1. When either the M or the S curve appears to fit poorly, an FP of higher degree may be tried. Figures 7 and 8 show running line smooths of `Z_ml` and `Z_ml` squared. There are no obvious systematic trends or peculiarities, and we may conclude informally from the plots that the M and S curves are satisfactory.

If the normal model appears to fit well, the centiles may be back transformed to the original scale and the resulting median and reference interval plotted as in Figure 1 for the fetal kidney volume data.

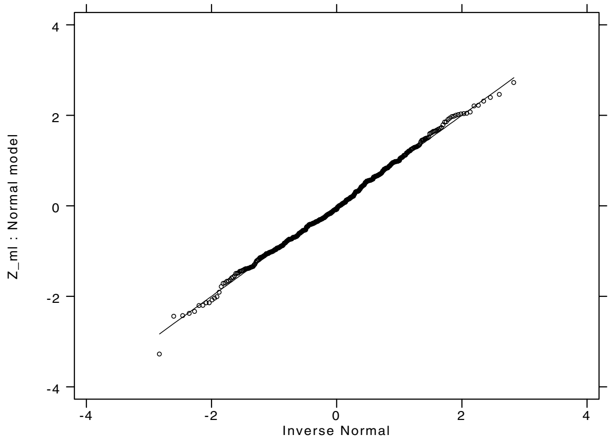


Figure 6. Normal plot of Z_ml for kidney volume data.

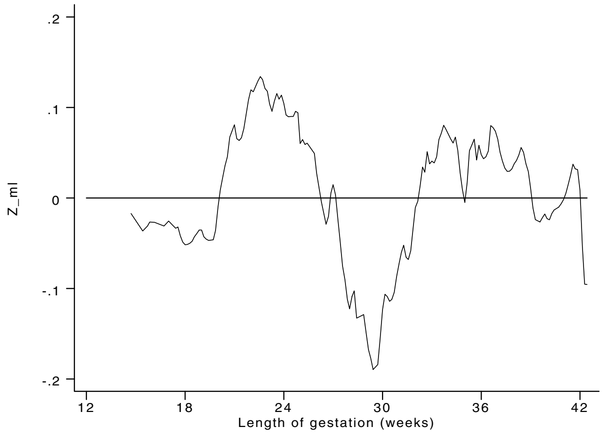


Figure 7. Local linear smooth of Z_ml vs. age.

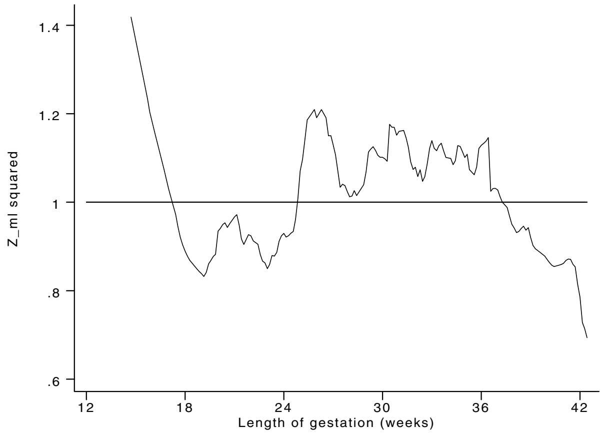


Figure 8. Local linear smooth of Z_ml squared vs. age.

Exponential normal models

A fitted normal model with non-normal skewness or kurtosis in its standardized (Z) values will lead to inaccurate centile estimates. To accommodate skewness, an exponential normal (EN) model may be fitted (Manly 1976). The amount of skewness is controlled by a parameter called γ_T. Non-normal kurtosis may be coped with using an extension of the EN model called the MEN model, which involves a so-called modulus transformation (John and Draper 1980) with a parameter δ_T. The sequence of models—normal, EN and MEN—have standardized values which may be written respectively as

$$Z = \frac{Y - \mu_T}{\sigma_T}$$

$$U = \frac{\exp(\gamma_T Z) - 1}{\gamma_T}$$

$$V = \operatorname{sign}(U) \frac{1}{\delta_T} \left[(1 + |U|)^{\delta_T} - 1 \right]$$

Expressions for the centile estimates for the EN and MEN distributions may be found by algebraically inverting these formulae. For the EN and MEN distributions, μ_T and σ_T are no longer the mean and SD but are the median and a standard deviation-like scale parameter. Greater mathematical detail of these models is given in Royston (1996) and Royston and Wright (1996).

The parameters of the distributions are essentially independent and so the FP powers of T selected for the M and S curve in the normal model (as described above) are suitable for the EN and MEN models also. The γ_T and δ_T shape parameters are referred to as G and D curves. To investigate non-normal skewness, we recommend fitting an EN model with a constant estimated value of γ_T and the chosen powers of M and S, and comparing the deviance with that of the corresponding normal model. To assess non-normal kurtosis, the deviance of the MEN model with constant δ_T should be compared with that of the EN model. Goodness-of-fit tests may still reveal non-normality in the standardized values. In these situations, further modeling of the γ_T and δ_T parameters may be required and we suggest fitting linear, then quadratic terms for them. Although important age variations in skewness and kurtosis are not common in practice, when either does exist simple polynomials are usually adequate models for them (Royston and Wright 1996).

To illustrate the use of these more elaborate models, we consider the American HANES I Survey of Health and Nutrition (Hamill et al. 1977). As part of the survey, the heights of 2274 girls aged between 1 and 11 years were recorded (see Figure 9).

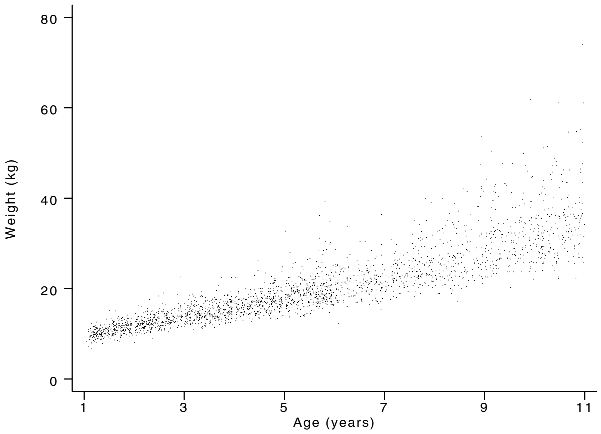


Figure 9. Weight vs. age.

Fractional polynomial regression of log(height) on age suggests suitable powers for the M and S curves to be $(-2, 1)$ and (1) respectively. The deviance obtained from `xriml` for the normal model is -2366.69 and the Shapiro Wilk test applied to the corresponding standardized values (`Z_ml`) has $P < 0.0001$. The EN model is a significant improvement on the normal model; the deviance is reduced by 75.49. The Shapiro Wilk test ($P = 0.0006$) still indicates non-normality. The MEN model is an even better fit than the EN model, the deviance reduction being 11.50, and a W test of the standardized values shows no significant non-normality ($P = 0.12$). A normal plot for `Z_ml` for this model is given in Figure 10.

The output for the final model is as follows:

```
. xriml loght age, fp(m:-2 1,s:1) di(men)

Iteration 0:  Log Likelihood =  1190.8117
Iteration 1:  Log Likelihood =  1224.5177
Iteration 2:  Log Likelihood =  1226.8225
Iteration 3:  Log Likelihood =  1226.8414
Iteration 4:  Log Likelihood =  1226.8419

Modulus exp-Normal Regression                      Number of obs    =     2273

Log Likelihood =   1226.8418577
```

```
------------------------------------------------------------------------------
   loght |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
Mcurve   |
    Xm_1 |  -.2494223   .0239616    -10.409   0.000      -.2963861   -.2024586
    Xm_2 |   .1154281   .0016735     68.975   0.000       .1121481     .118708
   _cons |   2.308695   .0105783    218.248   0.000       2.287962    2.329428
---------+--------------------------------------------------------------------
Scurve   |
    Xs_1 |   .0082277   .0007748     10.619   0.000       .0067091    .0097463
   _cons |   .0832634   .0046861     17.768   0.000       .0740788    .0924481
---------+--------------------------------------------------------------------
Gcurve   |
   _cons |   -.121107   .0163471     -7.408   0.000      -.1531466   -.0890673
---------+--------------------------------------------------------------------
Dcurve   |
   _cons |   .8201514   .0505691     16.218   0.000       .7210377     .919265
------------------------------------------------------------------------------
  Final deviance = -2453.684 (2273 observations.)
```

Note that any parameter curves for a given distribution which do not have specified powers of age in the `fp` option will be estimated by a constant, e.g. G and D are constants in the MEN model fitted above.

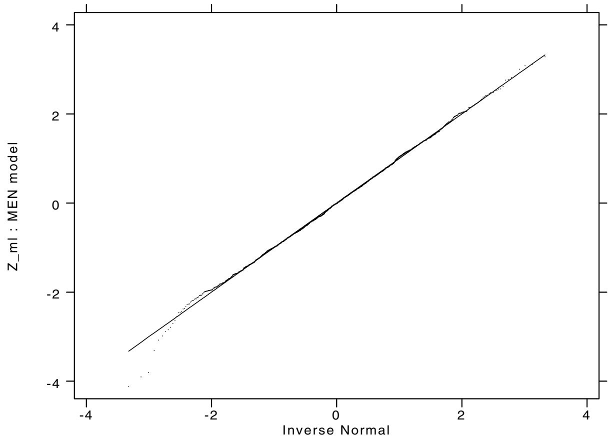


Figure 10. Normal plot of Z_ml for weight data.

In some contexts, such as assessment of child size, the aim is to provide estimates of a range of centiles and not just a median and reference interval. In Figure 11, the 0.5th, 3rd, 10th, 50th, 90th, 97th, and 99.5th centiles on the original scale are plotted for the final model described above. Centiles may be calculated in `xriml` using the option `centile(0.5 3 10 50 90 97 99.5)`. An alternative is the `centcalc` routine, which requires variables containing the estimated parameters but needs no further maximum likelihood fitting. The command in this example is as follows :

```
. centcalc M_ml S_ml, gamma(G_ml) delta(D_ml) di(men) cent(0.5 3 10 50 90 97 99.5)
```

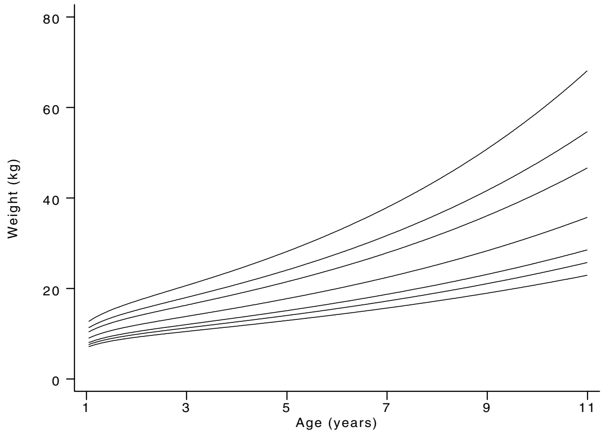


Figure 11. 0.5, 3, 10, 50, 90, 97, 99.5 centiles for weight vs. age.

Comparison with qreg

Stata's `qreg` command performs parametric quantile regression with an estimation procedure based on the $L1$-norm. For a given centile (or quantile), a linear equation is estimated for the explanatory variable in terms of one or more covariates. Since this is an alternative to the method proposed here, a brief comparison is made using the fetal kidney volume data.

Quantile regression models for the 3rd, 50th, and 97th centiles of log kidney volume were fitted using `qreg` with the logarithm and square root of gestational age (found to be suitable FP powers above) as covariates. The resulting curves and those obtained from `xriml` are shown in Figures 12 and 13—the former figure illustrates the curves on the log scale and the latter on the original scale. (The full lines denote the `xriml` fit and the broken lines the `qreg` fit.) There is little difference between the two sets of curves, with the larger discrepancies at the higher gestational ages.

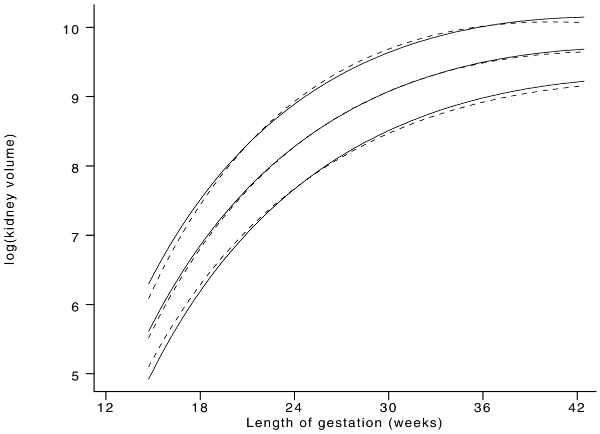


Figure 12. 94% RI and median for log(kidney volume) (full line: xriml, dotted line: qreg).

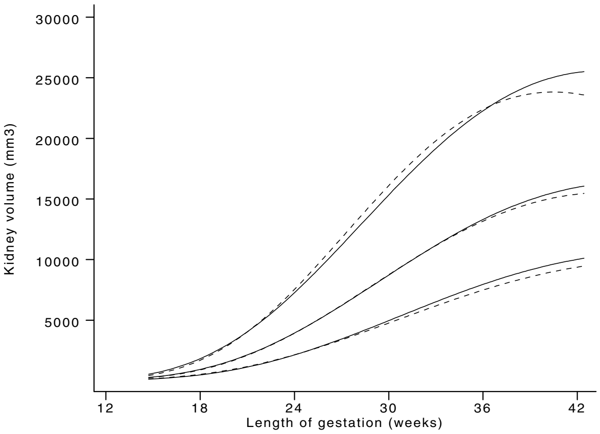


Figure 13. 94% RI and median for kidney volume (full line: xriml, dotted line: qreg).

The formulae for SD-scores and arbitrary centile estimates obtainable from the parametric approach in `xriml` are useful, for example when reporting the results for others to use. This feature is not available with `qreg`, though of course equations may be found for centile curves individually.

Extensions

The fitting of the normal, EN, and MEN distributions is described above. However, the `dist` option in `xriml` allows the fitting of a further three models—the three-parameter shifted-log-normal (SL) and power-normal (PN) distributions and the four-parameter modulus-power-normal (MPN) distribution. The basic principle is the same—the parameters are related to the median, standard deviation, skewness, and, in the four-parameter model, the kurtosis. Royston (1991) estimated the mean using least squares, then if necessary modeled non-normality using an origin-shifted logarithmic transformation. With `xriml`, a similar model may be fitted more conveniently by maximum likelihood in a single step.

Models based on the EN distribution for $\log Y$ are closely related to LMS models (Cole 1988 and Cole and Creen 1992) in which Y is assumed to follow a power-normal (PN) or Box–Cox power transformation model. The main distinction is that the median of Y is modeled on the original scale in the LMS approach but on a log scale in the EN model for $\log Y$. To mimic the LMS method, `dist(pn)` should be used in conjunction with the `cv` option, for which the S curve is a coefficient of variation (standard deviation divided by the median). An extension of the PN model which involves a modulus transformation and an additional shape parameter is called the MPN model (analogous to the relation between the EN and MEN models). Note that the PN and MPN models may be used only with data which are positively-valued—this restriction does not apply to the other models. Further details of the distributions and the relationships between them are given by Royston (1996).

The measurement of interest may depend on covariates in addition to age. The simplest case is that of a binary covariate, such as sex. Adding the option `covar(m:sex)` to an `xriml` command will add a term for `sex` to the M curve. Covariates may be included for the S, G, and/or D curves in similar fashion. Note however that the parameter curves for the two sexes are assumed parallel, which may be unrealistic. To fit curves with different shapes or slopes, new variables which contain suitable FP powers of age for males and females separately must be generated and included as covariates for the appropriate parameter curve.

The `se` option calculates approximate standard errors for the parameter and centile estimates. This allows calculation of confidence limits which indicate the precision of the centile estimates.

Syntax of the xriml command

`xriml` *yvar* [*xvar*] [`if` *exp*] [`in` *range*] `, dist(`*distribution_code*`)` [*major_options minor_options*]

Options

`dist(`*distribution_code*`)` is not optional. Valid *distribution_code*s are normal (`n`), exponential-normal (`en`), modulus exponential-normal (`men`), power-normal (or Box – Cox) (`pn`), modulus power-normal (`mpn`), and shifted log-normal (`sl`).

The *major_options* (most used options) are

`centile(`*#* [*#* [*#* ...]]`)` `fp(`[`m:`*term*`,`] [`s:`*term*`,`] [`g:`*term*`,`] [`d:`*term*]`)`

The *minor_options* (less used options), in alphabetic order, are

`covars(`*covar_list*`)` `cv` `init(`[`g:`*#*`,`] [`d:`*#*]`)` `ltolerance(`*#*`)` `nograph`
`noscaling` `saving(`*filename*[`, replace`]`)` `se`

Major options

`centile(`*#* [*#* [*#* ...]]`)` defines the centiles of *yvar* condition on *xvar* required, separated by commas or spaces. Default is 3 and 97; i.e., a 94% reference interval.

`fp(`[`m:`*term*`,`] [`s:`*term*`,`] [`g:`*term*`,`] [`d:`*term*]`)` specifies the fractional polynomial power(s) in *xvar* for the M, S, G, and D regression models. *term* is of the form [`powers`] *#* [*#* ...] or `fix` *#*. The word `powers` is optional. The powers should be separated by spaces, for example `fp(m:powers 0 1,s:powers 2)`, or equivalently `fp(m:0 1,s:2)`. If `powers` or `fix` are not specified for any curve, the curve is assumed to be a constant (`_cons`) estimated from the data. `fix` *#* implies that the corresponding curve is not to be estimated from the data, but is to be fixed at *#*. `fix` is valid only with the `g:` and `d:` parameters. Default is constants for each curve.

Minor options

`covars(`[`m:`*mcovars*`,`] [`s:`*scovars*`,`] [`g:`*gcovars*`,`] [`d:`*dcovars*]`)` includes *mcovars* (*scovars*, *gcovars*, *dcovars*) as predictors in the regression model for the M (S, G, D, if applicable) curves.

`cv` parameterizes the S curve to be a coefficient of variation (CV, standard deviation divided by median), rather than a standard deviation.

`init([g:#,] [d:#])` specifies initial values for the G (`g:`) and D (`d:`) curves. Defaults are shown below.

`dist`	Default	
code	G	D
`n`	N/A	N/A
`sl`	0	N/A
`pn`	1	N/A
`en`	0.01	N/A
`mpn`	1	1
`men`	0.2	1

`ltolerance(#)` is a convergence criterion for the iterative fitting process. For convergence, the difference between the final two values of the log-likelihood must be less than `ltolerance`. Default is 0.001.

`nograph` suppresses the default plot of *yvar* against *xvar* with fitted median and reference limits.

`noscaling` suppresses automatic scaling of *xvar* and its powers (see [R] **fracpoly**).

`saving(`*filename*`[, replace])` saves the graph to a file (see `nograph` option above).

`se` produces standard errors of the M and S (and G and D, if applicable) curves. Standard errors of the estimated reference limits are also calculated. (Warning: this option is computationally intensive when determining SEs of centiles, and may take considerable time on a slow computer and/or with a large dataset.)

Saved Results

`xriml` saves the deviance ($-2 \times$ log-likelihood of the final model) in `S_1`.

Syntax of the centcalc command

```
centcalc mvar svar [if exp] [in range] [, centiles(cent_list) dist(n|en|men|pn|mpn|sl)
                                        gamma(#|gvar) delta(#|dvar) cv prefix(stub) ]
```

Options

`centiles(`*cent_list*`)` is the list of estimated centiles (e.g. 50 for the median) to be calculated. Values in *cent_list* must be separated by commas and/or spaces.

`dist(n|en|men|pn|mpn|sl)` determines the distributional model (see help `xriml` for further details). If `dist()` is not specified, it is taken from the macro `S_dist`.

`gamma(`*#|gvar*`)` defines the (first) shape parameter as a variable (*gvar*) or as a number (*#*). If `gamma()` is not specified, it is taken from the macro `S_gamma`.

`delta(`*#|dvar*`)` defines the second shape parameter as a variable (*dvar*) or as a number (*#*). If `delta()` is not specified, it is taken from the macro `S_delta`.

`cv` is an option relating to the parameterization of the S curve for the chosen distribution. See `xriml` for more information.

`prefix()` is the prefix of the names of the new variables created to hold the estimated centiles, each labelled appropriately (for example `C95` for the 95th centile). Default is `C`.

Acknowledgment

This research received financial support from project grant number 039911/Z/93/Z from The Wellcome Trust.

References

Altman, D. G. 1993. Construction of age-related reference centiles using absolute residuals. *Statistics in Medicine* 12: 917–924.

Altman, D. G. and L. S. Chitty. 1993. Design and analysis of studies to derive charts of fetal size. *Ultrasound in Obstetrics and Gynecology* 3: 378–383.

Cole, T. J. 1988. Fitting smoothed centile curves to reference data (with discussion). *Journal of the Royal Statistical Society, Series A* 151: 385–418.

Cole, T. J. and P. J. Green. 1992. Smoothing reference centile curves: The LMS method and penalized likelihood. *Statistics in Medicine* 11: 1305–1319.

Hamill, P. V. V., T. A. Drizd, C. L. Johnson, R. B. Reed, and A. F. Roche. 1977. NCHS growth curves for children birth–18 years. *Washington DC: National Center for Health Statistics Vital and Health Series 11*.

John, J. A. and N. R. Draper. 1980. An alternative family of transformations. *Applied Statistics* 29: 190–197.

Manly, B. F. J. 1976. Exponential data transformations. *Statistician* 25: 37–42.

Mann, J. I., B. Lewis, J. Shepherd, A. F. Winder, S. Fenster, L. Rose, and B. Morgan. 1988. Blood lipid concentrations and other cardiovascular risk factors: distribution, prevalence and detection in Britain. *British Medical Journal* 296: 1702–1706.

Royston, P. 1991. Constructing time-specific reference ranges. *Statistics in Medicine* 10: 675–690.

Royston, P. 1996. Parametric models for estimating reference intervals in medicine. Technical Report TR-96-07. Statistics Section, Department of Mathematics, Imperial College.

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

Royston, P. and E. M. Wright. 1996. A parametric method for estimating age-specific reference intervals (“normal ranges”). *Journal of the Royal Statistical Society Series A*, submitted.

Sasieni, P. 1995. sed9: Symmetric nearest neighbor linear smoothers. *Stata Technical Bulletin* 24: 10–14.

Wright, E. M. and P. Royston. 1996. A comparison of statistical methods for age-related reference intervals. *Journal of the Royal Statistical Society Series A*, in press.

smv3.1 Discriminant analysis: An enhanced command

Joseph Hilbe, Arizona State University, atjmh@asuvm.inre.asu.edu

A first version of `discrim`, a program to model discriminant analysis when the response or grouping variable is 0/1 dichotomous, was published in the January 1992 issue of the STB, see Hilbe (1992). Since then it has apparently enjoyed good use from those who knew of its existence. However, the code is now outdated and suggested enhancements have been proposed — all of which have been incorporated in this revision.

The manner in which the discriminant estimation is performed in `discrim` is quite unique. The details of the mathematics involved are discussed in the original article. Selvin (1995) has presented an excellent overview of discriminant analysis using the first version of `discrim` in his recent text. I recommend it to anyone using the `discrim` command.

Syntax

`discrim` *grpvar* [*varlist*] [`if` *exp*] [`in` *range*] [`, predict anova graph detail keep`]

Options

`predict` provides a confusion matrix of actual vs. predicted group cell counts. Also included is a listing of the following percentages: (1) correctly predicted, (2) model sensitivity, (3) model specificity, (4) false positive, and (5) false negative.

`anova` provides an ANOVA of discriminant scores vs. the group variable. Bartlett’s test for equal variances is included.

`graph` provides a classification graph showing correctly and incorrectly classified cases.

`detail` creates and lists the following for each retained case:

1.	Group	actual group value
2.	`PRED`	predicted group value
3.	`DIFF`	a star indicating misclassified cases
4.	`LnProb1`	logistic probability of grouyp 1 membership
5.	`DscIndex`	discriminant index
6.	`DscScore`	discriminant score

`keep` keeps the variables created in the `detail` option in memory. Since the program drops cases with missing predictor values, care must be taken if overwriting the original data set.

Saved Results

The standard output includes a listing of various summary statistics together with a display of model discriminant function and unstandardized coefficients. The summary statistics are saved in `S_E_` macros to allow the user to have access to them for subsequent analysis if necessary. A listing of created macros are listed as follows:

S_E_obs	number of observations	S_E_var	independent variables
S_E_ob0	observations in group==0	S_E_ob1	observations in group==1
S_E_cn0	centroid 0	S_E_cn1	centroid 1
S_E_cng	grand centroid	S_E_r2	R-squared
S_E_mah	Mahalanobis	S_E_eig	eigenvalue
S_E_cc	canonical correlation	S_E_e2	eta squared
S_E_lam	Wilk's lambda	S_E_chi	chi-squared

Example

For Stata's `auto.dta`,

```
   . discrim foreign price mpg weight length turn, anova predict detail
                    Dichotomous Discriminant Analysis
 Observations    = 74                            Obs Group 0 =        52
 Indep variables = 5                             Obs Group 1 =        22

 Centroid 0  =   -0.7503                         R-square    =    0.5776
 Centroid 1  =    1.7734                         Mahalanobis =    6.3689
 Grand Cntd  =    1.0231

 Eigenvalue   =    1.3675                        Wilk´s Lambda =  0.4224
 Canon. Corr. =    0.7600                        Chi-square    = 59.8976
 Eta Squared  =    0.5776                        Sign Chi2     =  0.0000
                          Discrim Function    Unstandardized
           Variable         Coefficients       Coefficients
           -------------------------------------------------
           price              -0.0008               0.0003
           mpg                 0.1624              -0.0643
           weight              0.0072              -0.0029
           length             -0.0806               0.0320
           turn                0.3136              -0.1243
           constant          -16.0869               6.8860
                         ----- Predicted -----
             Actual   |  Group 0        Group 1 |   Total
             ---------+-------------------------+--------
             Group 0  |       43           9    |      52
             Group 1  |        0          22    |      22
             ---------+-------------------------+--------
             Total    |       43          31    |      74
             ---------+-------------------------+--------
                     Correctly predicted =  87.84 %
                     Model sensitivity   =  82.69 %
                     Model specificity   = 100.00 %
                     False positive      =   0.00 %
                     False negative      =  29.03 %
                  Discriminant Scores v Group Variable
                         Analysis of Variance
     Source              SS         df      MS            F     Prob > F
 ------------------------------------------------------------------------
 Between groups      98.4602843      1   98.4602843     98.46     0.0000
  Within groups      72.0000103     72   1.00000014
 ------------------------------------------------------------------------
     Total           170.460295     73   2.33507253
 Bartlett´s test for equal variances:  chi2(1) =  11.3597  Prob>chi2 = 0.000
  PRED    = Predicted Group       DIFF     = Misclassification
  LnProb1 = Probability Gr 1      DscScore = Discriminant Score
                                  DscIndex = Discriminant Index
 ---------------------------------------------------------------
       foreign       PRED  DIFF    LnProb1   DscIndex   DscScore
   1.        0          0          0.0592     2.7660    -0.5845
   2.        0          0          0.0041     5.4924    -1.6648
   3.        0          0          0.3090     0.8049     0.1926
   4.        0          0          0.0341     3.3448    -0.8138
   5.        0          0          0.0078     4.8420    -1.4071
   6.        0          0          0.0121     4.4051    -1.2340
   7.        0          1    *     0.9267    -2.5374     1.5170
   8.        0          0          0.0279     3.5520    -0.8959
   9.        0         0        0.0677     2.6220    -0.5274
  10.        0         0        0.0055     5.1908    -1.5453
  11.        0         0        0.0202     3.8826    -1.0269
  12.        0         1   *    0.6891    -0.7961     0.8270
  13.        0         0        0.0689     2.6032    -0.5200
  14.        0         1   *    0.7991    -1.3808     1.0587
  15.        0         0        0.0083     4.7789    -1.3821
 (output omitted)
  73.        1         1        0.9910    -4.7000     2.3739
  74.        1         1        0.9887    -4.4718     2.2835
```

References

Hilbe, J. 1992. smv3: Regression-based dichotomous discriminant analysis. *Stata Technical Bulletin* 5: 13–17.

Selvin, S. 1995. *Practical Biostatistical Methods*. Belmont, CA: Duxbury.

sts12 | A periodogram-based test for white noise

H. Joseph Newton, Texas A&M University, FAX (409) 845-3144, jnewton@stat.tamu.edu

Given a time series data set $X_1, \ldots, X_n$, one of the first things an analyst should do is to test the null hypothesis that the data come from a white noise process of uncorrelated random variables having a constant mean and constant variance. One common method for doing this test is referred to as Bartlett's test (see Newton (1988, 172) and Bartlett (1955, 92–94) for details) and consists of the following steps:

1. Calculate the periodogram of the data set, that is

$$\hat{f}(\omega_j) = \frac{1}{n}\left|\sum_{t=1}^{n}\left(X_t - \bar{X}\right) e^{2\pi i(t-1)\omega_j}\right|^2$$

   at the frequencies $\omega_j = (j-1)/n$, $j = 1, \ldots, q = [n/2]+1$. Under the null hypothesis of white noise, except for the values at frequencies 0 and 0.5, these values should look like a random sample from a constant multiple of a χ^2_2 distribution ($\hat{f}(0) = 0$ and $\hat{f}(0.5)$ is a multiple of a χ^2_1).

2. From the periodogram calculate the cumulative periodogram

$$\hat{F}(\omega_k) = \frac{\sum_{j=1}^{k}\hat{f}(\omega_j)}{\sum_{j=1}^{q}\hat{f}(\omega_j)}, \qquad k = 1, \ldots, q$$

   Note that $\hat{F}(0) = 0$, $\hat{F}(\omega_q) = 1$ and under white noise, a plot of the cumulative periodogram versus frequency should fall randomly along a line from (0,0) to (0.5,1).

3. To measure the deviation of $\hat{F}$ from the expected straight line, we calculate

$$B = \sqrt{q}\max_{1\le k\le q}\left|\hat{F}(\omega_k) - \frac{k}{q}\right|$$

   Under white noise,

$$\lim_{n\to\infty}\Pr(B \le b) = \sum_{j=-\infty}^{\infty}(-1)^j e^{-2b^2j^2} \equiv G(b)$$

   Thus, the null hypothesis of white noise is rejected if the calculated value of B leads to a p-value calculated from the cdf G less than a specified α.

This test is often presented graphically as follows. On a plot of the cumulative periodogram, one can place the line k/q as well as parallel lines on either side of this center line at a distance equal to the value of B having $G(B) = 1-\alpha$. Then white noise is rejected if and only if the plot of $\hat{F}$ crosses either of the two parallel lines.

In this insert we present a program called `wntestf` which performs this analysis as well as the programs `bartcdf` and `bartq` which calculate the cdf and quantile functions of B under the hypothesis of white noise.

Syntax of the wntestf command

`wntestf` *varname* [`if` *exp*] [`in` *range*] [, `nograph` `level(`*#*`)` `t(`*varname*$_t$`)` `xlabel(`*#*,...,*#*`)` `ylabel(`*#*,...,*#*`)` `saving(`*filename*`)` `t1title(`*string*`)` *graph_options*]

Options

`nograph` suppresses drawing the graph.

`level(`*#*`)` specifies the confidence level, in percent, for the confidence bands in the graph if the graph is to be drawn. The default is `level(95)` or as set by `set level`; see [U] **26.4 Specifying the width of confidence intervals**.

`t(`*varname*$_t$`)` specifies the variable name that contains the time at which the observation was recorded. This must be specified once as the variable named here is used to determine the order of the observations in the estimation. Once specified, however, it need not be specified again except to change the variables identity. Note that the data must be equally spaced in time.

`xlabel(`*#*,...,*#*`)` is `graph, twoway`'s `xlabel()` option for labeling the x-axis. The default is `xlabel(0,.1,.2,.3,.4,.5)`; see [R] **graph axis labels**.

`ylabel(`*#*,...,*#*`)` is `graph, twoway`'s `ylabel()` option for labeling the y-axis. The default is `ylabel(0,.2,.4,.6,.8,1)`; see [R] **graph axis labels**.

`saving(`*string*`)` is `graph`'s `saving()` option; see [R] **graph saving**.

`t1title(`*string*`)` is `graph, twoway`'s `t1title()` option for specifying the `t1` title. The default title is "Cumulative Periodogram White Noise Test"; see [R] **graph titles**.

graph_options are any of the other options allowed with `graph, twoway`; see [R] **graph twoway**.

Saved Results

The values of B and the p-value $1 - G(B)$ of the test are placed in the global macros `S_1` and `S_2`, respectively.

Syntax for the bartcdf and bartq commands

`bartcdf` *#* calculates $G(\#)$ and places its value in the global macro `S_1`.

`bartq` *#* calculates the value q of the random variable B such that $G(q) = \#$ and places the result in the global macro `S_2`.

Examples

We start by using Stata's normal random number generator and testing the true null hypothesis of white noise without producing the graph:

```
. set obs 100
. set seed 123456
. gen x = invnorm(uniform())
. gen t = _n
. wntestf x, t(t) nogr

. disp_s
S_1:        .4599050096001969
S_2:        .9840305472233085
```

Thus the value of B is 0.4599 which has a p-value of 0.9840, which means we would not reject the white noise hypothesis for this data set.

In Figure 1, we graph another Gaussian white noise series of 100 observations.

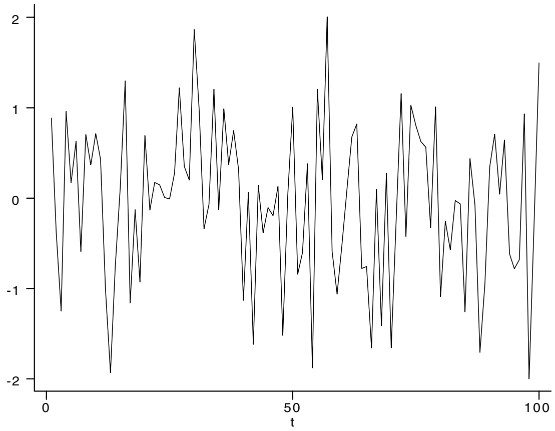


Figure 1. A normal white noise series of length 100.

In Figure 2, we have plotted a point plot of the 51 values of the periodogram of the data in Figure 1. The frequencies are $0, 1/100, 2/100, \ldots, 50/100 = 0.5$. As expected, the points look quite random with a few large values. This is what a random sample from a χ^2_2 population would look like.

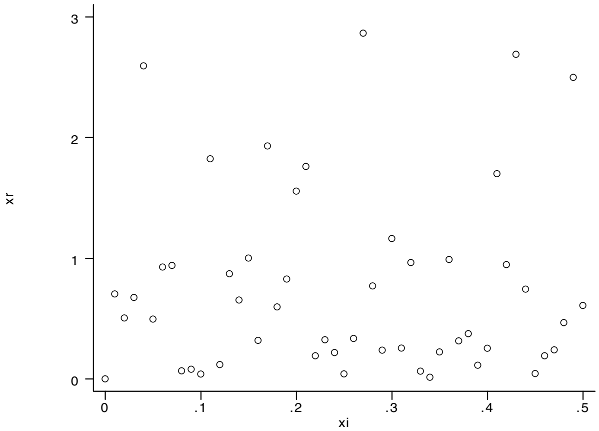


Figure 2. Periodogram of the data in Figure 1.

In Figure 3, we have the result of using the default 95% confidence level. The cumulative periodogram falls entirely within the confidence band and so again we don't reject the hypothesis of white noise.

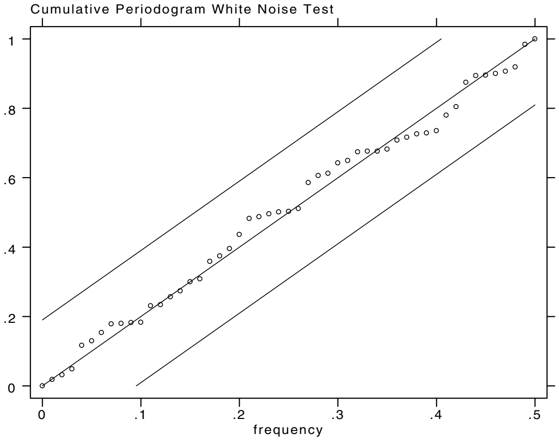


Figure 3. Result of `wntestf` for the data in Figure 1.

Finally, in Figure 4 we plot a non-white noise series of length $n = 200$ formed by

```
. gen x = invnorm(uniform()) + cos(2*_pi*_n/10)
```

This is a cosine curve (with amplitude one and period 10 time units) plus a white noise time series. Because the amplitude is small relative to the variation in the noise, it is not obvious from the data plot that there is a cosine curve embedded in the noise.

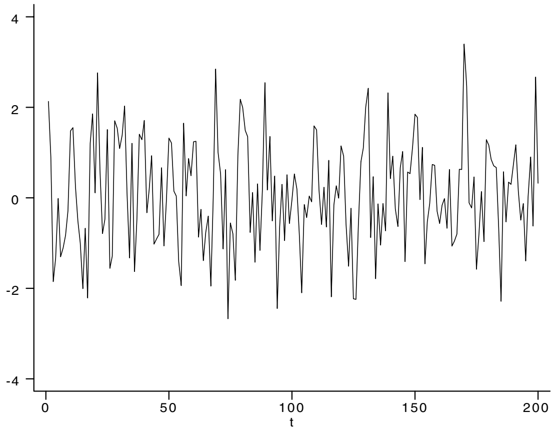


Figure 4. A normal white noise series plus a cosine of period 10 and amplitude 1.

That there is a cosine of period 10 is very obvious from Figure 5 where the result of using `wntestf` is displayed. Note the large jump in the cumulative periodogram at frequency 0.1 corresponding to the period 10, and that the confidence bands are crossed at this point, thus leading to rejection of white noise.

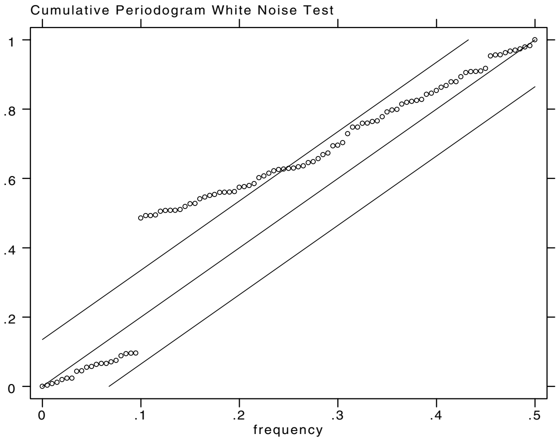


Figure 5. Result of `wntestf` for the data in Figure 4

Remarks

`wntestf` uses several of the graphics primitives in data coordinates described in the insert *gr20* described elsewhere in this issue of the STB.

References

Bartlett, M. S. 1955. *An Introduction to Stochastic Processes with Special Reference to methods and Applications*. Cambridge: Cambridge University Press.

Newton, H. J. 1988. *TIMESLAB: A Time Series Analysis Laboratory*. Pacific Grove, California: Wadsworth & Brooks/Cole.

STB categories and insert codes

Inserts in the STB are presently categorized as follows:

General Categories:

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

Statistical Categories:

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

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

International Stata Distributors

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

Company: Applied Statistics &
Systems Consultants
Address: P.O. Box 1169
Nazerath-Ellit 17100, Israel
Phone: +972 6554254
Fax: +972 6554254
Email: sasconsl@actcom.co.il
Countries served: Israel

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

Company: Metrika Consulting
Address: Roslagsgatan 15
113 55 Stockholm
Sweden
Phone: +46-708-163128
Fax: +46-8-6122383
Email: hedstrom@metrika.se
Countries served: Baltic States, Denmark, Finland,
Iceland, Norway, Sweden

Company: Ritme Informatique
Address: 34 boulevard Haussmann
75009 Paris
France
Phone: +33 1 42 46 00 42
Fax: +33 1 42 46 00 33
Email: ritme.inf@applelink.apple.com
Countries served: Belgium, France,
Luxembourg, Switzerland

Company: Smit Consult
Address: Scheidingstraat 1
Postbox 220
5150 AE Drunen
Netherlands
Phone: +31 416-378 125
Fax: +31 416-378 385
Email: j.a.c.m.smit@smitcon.nl
Countries served: Netherlands

Company: Timberlake Consultants
Address: 47 Hartfield Crescent
West Wickham
Kent BR4 9DW U.K.
Phone: +44 181 462 0495
Fax: +44 181 462 0493
Email: 100412.2603@compuserve.com
Countries served: Ireland, U.K.

Company: Timberlake Consultants
Satellite Office
Address: Praceta do Comércio,
N°13–9° Dto. Quinta Grande
2720 Alfragide Portugal
Phone: +351 (01) 4719337
Telemóvel: 0931 62 7255
Email: 100412.2603@compuserve.com
Countries served: Portugal