# STATA TECHNICAL BULLETIN

July 1996
STB-32

A publication to promote communication among Stata users

## Contents of this issue

## an61 Submission guidelines

H. Joseph Newton, Stata Technical Bulletin, stb@stata.com

### The text of the insert

Please send the text of your submission in a plain ASCII file. Don't worry about formatting the document. I have to change everything to conform to our electronic publishing standards anyway. If your document contains mathematical symbols or graphs, FAX (409-845-3144) or mail me copies so I can see the intended appearance of the text. If you have used STAGE to edit the graphs in your submission, be sure to include the `.gph` files. Do not add titles (e.g., "Figure 1: ....") to your graphs, as I will have to strip them off.

### The software (if any)

Send the ado-files, `.exe` files, or other software that accompanies your submission.

### Example with datasets

Include a do-file that replicates the examples in your text. Also include the datasets used in the example. This allows me to verify that the software works as described and allows users to replicate the example as a way of learning how to use your software.

### Help files

Each ado-file needs a help file. Look at an example on a recent STB diskette for the structure of the help file. If you have questions, just fill in as much of the information as possible, and I will take care of the details.

### Format of email submissions

If you are on a Unix system or have access to Unix utilities, it is easy to submit your insert article via electronic mail. Many of the modern mail programs on other platforms have the ability to attach binary files to a message. The safest thing to do if you have any question is to send me an email message telling me what platform you are using and we can work together on how to get the submission to me electronically. If nothing else, you can mail me a diskette.

The following instructions are for electronic submission of STB inserts from an account on a Unix computer:

### Instructions for submitting STB inserts via electronic mail

There are three steps for accumulating the material into a mail message: (1) use the Unix `tar` command to store all the individual files in a single archive; (2) use the Unix `compress` command to reduce the size of the `.tar` file; (3) use the Unix `uuencode` command to convert binary information to a form that can be mailed reliably.

### To send a few files

As an example, imagine that you wish to send files named `abc.def` and `ghi.jkl`. The following command will work

```
tar -cf - abc.def ghi.jkl | compress | uuencode xyzzy.tar.Z > whatever
```

This command will create a file named `whatever` in the current directory. Include this file in your email.

### To send the current directory and all of its subdirectories

To send the current directory and all of its subdirectories, type

```
tar -cf - . | compress | uuencode xyzzy.tar.Z > whatever
```

Again, the file named `whatever` contains the desired information. This is a good approach for the STB. You can store ado-files in one subdirectory, text in another, data in another, graphs in another, and so on.

Be aware of the size of the `whatever` file. Some mail systems cannot handle files larger than 100,000 bytes. In either approach, it is best if you include a text file with an annotated list of the files you are sending.

Finally, you can send the `whatever` file using the command

```
mail -s "STB insert" stb@stata.com < whatever
```

where `stb@stata.com` is the official electronic mailing address for the *Stata Technical Bulletin*.

## crc44 Confidence intervals in a $2 \times 2$ table

Peter Sasieni, Imperial Cancer Research Fund, London, FAX (011)-44-171-269-3429, sasieni@icrf.icnet.uk

The commands `cc`, `cci`, `cs` and `csi` from the `epitab` group of programs all produce confidence intervals for the odds ratio in a 2×2 table. The default method for calculating the confidence intervals is that proposed by Cornfield and by Gart. As far as I am aware the method is valid even when there is a zero in the 2×2 table. It is however generally anti-conservative compared to the exact method and will be particularly poor in situations in which the usual chi-squared test for independence is a poor approximation to the Fisher exact test. I recommend only using the Cornfield confidence interval when the expected frequency (under independence) in all four cells of the 2×2 table is at least 5.

The `epitab` commands do not attempt to calculate a confidence interval whenever there is a zero in the table. In such instances one would like a one-sided confidence interval. Although one might argue that exact confidence intervals are relatively easy to calculate when there is a zero in a 2×2 table, these are not yet available in Stata and the enclosed modification to `_crcor.ado` and `_crcrnfd.ado` provide a quick fix.

There was also a bug in the old versions of these programs, so that occasionally one could get a negative limit to the confidence interval for the odds ratio. This is fixed in these files.

### Example 1

The old version gave

```
. cci 2 224 1 832
                                                         Proportion
                 |   Exposed   Unexposed  |      Total     Exposed
-----------------+------------------------+----------------------
           Cases |         2         224  |        226      0.0088
        Controls |         1         832  |        833      0.0012
-----------------+------------------------+----------------------
           Total |         3        1056  |       1059      0.0028
                 |                        |
                 |      Pt. Est.          |  [95% Conf. Interval]
                 |------------------------+----------------------
      Odds ratio |      7.428571          |  .9671183   -4.945097  (Cornfield)
 Attr. frac. ex. |      .8653846          | -.0339996     1.20222  (Cornfield)
 Attr. frac. pop |      .0076583          |
                 +-----------------------------------------------
                              chi2(1) =     3.68  Pr>chi2 = 0.0550
```

The new version produces

```
. cci 2 224 1 832
                                                         Proportion
                 |   Exposed   Unexposed  |      Total     Exposed
-----------------+------------------------+----------------------
           Cases |         2         224  |        226      0.0088
        Controls |         1         832  |        833      0.0012
-----------------+------------------------+----------------------
           Total |         3        1056  |       1059      0.0028
                 |                        |
                 |      Pt. Est.          |  [95% Conf. Interval]
                 |------------------------+----------------------
      Odds ratio |      7.428571          |  .9671183           .  (Cornfield)
 Attr. frac. ex. |      .8653846          | -.0339996           .  (Cornfield)
 Attr. frac. pop |      .0076583          |
                 +-----------------------------------------------
                              chi2(1) =     3.68  Pr>chi2 = 0.0550
```

In fact neither of these are ideal since the exact 95% confidence interval is $[0.38, \infty)$. This difference is reflected in the difference between the chi-squared and the exact $p$-values. The exact $p$-value is 0.1169. This can be produced in Stata by using the exact option. Note that the expected number of exposed cases under independence is only 0.64. It is not surprising then that the asymptotic confidence interval is poor.

### Example 2

```
. cci 0 10 5 20
                                                         Proportion
                 |   Exposed   Unexposed  |      Total     Exposed
-----------------+------------------------+----------------------
           Cases |       0          10    |         10      0.0000
        Controls |       5          20    |         25      0.2000
-----------------+------------------------+----------------------
           Total |       5          30    |         35      0.1429
                 |                        |
                 |      Pt. Est.          |  [95% Conf. Interval]
                 |------------------------+----------------------
      Odds ratio |             0          |          0    1.742592  (Cornfield)
 Prev. frac. ex. |             1          | -.7425922           1  (Cornfield)
 Prev. frac. pop |             .          |
                 +-----------------------------------------------
                               chi2(1) =     2.33  Pr>chi2 = 0.1266
```

In this example, the exact interval is $[0, 2.675]$ and the exact (2-sided) $p$-value is 0.29. Again the expected number of exposed cases under independence is only 2.

```
. cci 0 100 20 180
                                                         Proportion
                 |   Exposed   Unexposed  |      Total     Exposed
-----------------+------------------------+----------------------
           Cases |       0         100    |        100      0.0000
        Controls |      20         180    |        200      0.1000
-----------------+------------------------+----------------------
           Total |      20         280    |        300      0.0667
                 |                        |
                 |      Pt. Est.          |  [95% Conf. Interval]
                 |------------------------+----------------------
      Odds ratio |             0          |          0    .3485028  (Cornfield)
 Prev. frac. ex. |             1          |  .6514972           1  (Cornfield)
 Prev. frac. pop |             .          |
                 +-----------------------------------------------
                               chi2(1) =    10.71  Pr>chi2 = 0.0011
```

In this example, the exact interval is similar $[0, 0.376]$ to the one given by Cornfield's method. The expected number of exposed cases is 6.7.

```
. cci 15 35 0 50
                                                         Proportion
                 |   Exposed   Unexposed  |      Total     Exposed
-----------------+------------------------+----------------------
           Cases |      15          35    |         50      0.3000
        Controls |       0          50    |         50      0.0000
-----------------+------------------------+----------------------
           Total |      15          85    |        100      0.1500
                 |                        |
                 |      Pt. Est.          |  [95% Conf. Interval]
                 |------------------------+----------------------
      Odds ratio |             .          |  5.403903           .  (Cornfield)
 Attr. frac. ex. |             1          |  .8149486           .  (Cornfield)
 Attr. frac. pop |            .3          |
                 +-----------------------------------------------
                               chi2(1) =    17.65  Pr>chi2 = 0.0000
```

This example shows what happens when the odds ratio is infinite. The exact interval is $[4.65, \infty)$. The expected number of exposed cases (and controls) is 7.5.

## Reference

Cornfield's limits are discussed by Breslow and Day (1980, 133–134). On page 133, they give a rule of thumb for when the approximation of Cornfield is adequate. On page 129, they give equations for the exact confidence limits. The whole topic is covered in Chapter 4.

Breslow, N. E. and N. E. Day. 1980. *Statistical Methods in Cancer Research*, vol. 1. Lyon: International Agency for Research on Cancer.

## dm42 Accrue statistics for a command across a by list

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

Many times users wish to aggregate statistics from a command that are collected across groups of the dataset. The usual method is to write a do-file and either collect the needed statistics by hand, or try to automate the collection via looping in a do-file where the statistics are collected in a new variable. The `accrue` command automates this job by collecting the statistics into a new dataset that can be further analyzed. Interested readers should also note Patrick Royston's earlier command `byvar`, which is easier to use in many cases.

There are still problems for which `accrue` is not suited. For instance, gathering together statistics from the `summarize` command is best done with the new `coll2` version of `collapse` since `summarize` will overwrite the result vector for each variable in the command line. The new `accrue` is suitable for gathering statistics from commands that do not write results for each variable in the command's *varlist*. It is suitable for those cases where you wish to issue a command with a `by` *varlist* and gather the statistics for each group in the `by` *varlist*.

The `accrue` command stores parts of the commands it needs in global macros until the `accrue exec` command is issued. It then reads in the parts of the command as they are needed and posts the results to a new `.dta` file. The `accrue using` command is used to specify the name of this file.

Additionally, `accrue` can be used to run an ado-command (that does not have a `by` option) across a `by` *varlist* since `accrue` does not require that you collect statistics. I highlight this alternative use in a later example.

## Syntax

The `accrue` command is actually a collection of commands that gathers together the needed information in the same manner that the `reshape` command gathers its information. These new commands are listed below with explanation.

`accrue cmd` *command*

This command specifies the Stata command that is to be run for all of the groups.

`accrue by` *varlist*

This is a list of existing variables that would normally appear in the `by` *varlist*`:` section of the command if you were to issue the command interactively.

`accrue using` *filename*

This specifies the filename into which to store the new `.dta` file of collected statistics. This should be a new filename unless the `replace` qualifier is used in the `accrue exec` command.

`accrue macros` *macroname-list*

This specifies the global macros that are to be collected for each run of the command.

`accrue mnames` *macroname-to-newvarname-list*

This is a list of names that you wish to use as the variable names for the macros that are listed in the `accrue macros` statement. These names will be used in the created dataset and there should be as many names here as there are macros that you wish to save.

`accrue results` *integer-result-vector-list*

This is a list of integers that correspond to the `_result()` vector entries that you wish to save.

`accrue rnames` *result-to-newvarname-list*

This is a list of names that you wish to use as the variable names for the `_result()`s that are listed in the `accrue results` statement. These names will be used in the created dataset and there should be as many names here as there are macros that you wish to save.

`accrue coeffs` *varlist*

This command specifies the variable names for which you would like to save the `_b[]` values. Note that this is required to be a variable list so that you cannot use `_cons` here to save the coefficient of the constant. In order to save the coefficient of the constant, you need to specify `_b[_cons]` as one of the arguments to `accrue macros`.

`accrue cnames` *coeffs-to-varname-list*

If you have specified variable names for `accrue coeffs` and do not specify names here, the variable names will be used. Otherwise, the coefficients that you requested will be saved in the corresponding names that you specify here. This list (if it exists) must be the same length as the list of variable names specified in the `accrue coeffs` command.

`accrue stderrs` *varlist*

This command specifies the variable names for which you would like to save the `_se[]` values. Note that this is required to be a variable list so that you cannot use `_cons` here to save the standard error of the constant. In order to save the standard error of the constant, you need to specify `_se[_cons]` as one of the arguments to the `accrue macros`.

`accrue snames` *standard-errors-to-varname-list*

If you have specified variable names for `accrue stderrs` and do not specify names here, the variable names will be used. Otherwise, the standard errors that you requested will be saved in the corresponding names that you specify here. This list (if it exists) must be the same length as the list of variable names specified in the `accrue stderrs` command. If you specify variables in `accrue coeffs` and `accrue stderrs`, then at least one of `accrue cnames` or `accrue snames` must be specified.

`accrue query`

This will show all of the parts of the `accrue` command that have been defined.

`accrue clear`

This will clear from memory all parts of the `accrue` command.

`accrue exec` [ `replace noisily nomsg` ]

This will perform the command outlined by all of the parts of the `accrue` command saving the results to the created dataset. If the `replace` option is included, then any previous results stored in the filename specified in the `accrue using` command will be replaced. The `noisily` option will allow you to see the results of the command as it is run for each of the groups. If the `nomsg` option is included, then the informatory messages showing which group is currently active will be suppressed. By default, `accrue` shows the current value of all of the `by` variables to keep you informed of the progress of the command.

Note that in all of the following examples, I limit myself to the small datasets that come with Stata. This command, however, is most useful when you have very large datasets and wish to collect a great deal of statistics. In those cases, the ability of Stata to specify *varlist*s with abbreviated notation makes the `accrue` command very powerful.

## Example 1

We begin with an example using the `auto.dta` dataset. In this example we want to collect the statistics from a regression where we model the price of a car by its weight, length, and mpg. We want to run this model for both domestic and foreign cars. For each of the models, we want to gather all of the statistics in the `_result()` vector and to get the coefficient and standard error of weight in the regression. Since we are just getting one coefficient and standard error, we will specify them in the `accrue macros` rather than individually listing them in the `accrue coeffs` and `accrue stderrs` commands.

```
. describe

Contains data from /usr/local/stata/auto.dta
  Obs:    74 (max= 20027)                    1978 Automobile Data
 Vars:    12 (max=    99)
Width:    44 (max=   200)
  1. make         str18  %18s                Make and Model
  2. price        int    %8.0g               Price
  3. mpg          int    %8.0g               Mileage (mpg)
  4. rep78        int    %8.0g               Repair Record 1978
  5. hdroom       float  %6.1f               Headroom (in.)
  6. trunk        int    %8.0g               Trunk space (cu. ft.)
  7. weight       int    %8.0g               Weight (lbs.)
  8. length       int    %8.0g               Length (in.)
  9. turn         int    %8.0g               Turn Circle (ft.)
 10. displ        int    %8.0g               Displacement (cu. in.)
 11. gratio       float  %6.2f               Gear Ratio
 12. foreign      int    %8.0g       foreign  Car type
Sorted by:  foreign

. accrue by foreign
. accrue using example
. accrue results 1 2 3 4 5 6 7 8 9
. accrue rnames nobs SSM dfM SSE dfE F Rsquare adjRsq rootmse
. accrue macros _b[weight] _se[weight]
. accrue mnames WGTcoef WGTse
. accrue cmd regress price weight length mpg
. accrue query

by:      foreign
macros:  _b[weight] _se[weight]
mnames:  WGTcoef WGTse
results: 1 2 3 4 5 6 7 8 9
rnames:  nobs SSM dfM SSE dfE F Rsquare adjRsq rootmse
using:   example
cmd:     regress price weight length mpg

. accrue exec replace nomsg
Posting results to: example

. use example
(1978 Automobile Data)
```

```
. list

Observation 1

     foreign            0     WGTcoef  6.767233466       WGTse   1.22632643
        nobs           52         SSM  271443324.9         dfM            3
         SSE  217751475.8         dfE           48           F  19.94518376
     Rsquare  .5548777799      adjRsq  .5270576411     rootmse  2129.903537

Observation 2

     foreign            1     WGTcoef  4.784841122       WGTse  1.670006309
        nobs           22         SSM  113515242.2         dfM            3
         SSE  30847970.56         dfE           18           F  22.07897119
     Rsquare  .7863169573      adjRsq  .7507031169     rootmse  1309.112731
```

Returning to the previous example, we can rerun the `accrue` command after adding another variable to the `by` *varlist*.

```
. use auto, clear
. replace rep78=10 if rep78==.
. accrue by foreign rep78
. accrue exec replace
Posting results to: example
foreign=0 rep78=1
foreign=0 rep78=2
foreign=0 rep78=3
foreign=0 rep78=4
foreign=0 rep78=5
foreign=0 rep78=10
foreign=1 rep78=3
foreign=1 rep78=4
foreign=1 rep78=5
foreign=1 rep78=10
   _b[weight] unavailable
   _se[weight] unavailable
```

Note that when `foreign` was equal to one and `rep78` was equal to 10, there was an informatory message printed by the `accrue` command. Returning to the dataset, we see that a regression cannot be computed for the `foreign==1 & rep78==10` group, since this group contains only one observation.

```
. reg price weight length if foreign==1 & rep78==10
insufficient observations
. list if foreign==1 & rep78==10
Observation 74

        make  Peugeot 604       price        12990         mpg           14
       rep78           10      hdroom          3.5       trunk           14
      weight         3420      length          192        turn           38
       displ          163      gratio         3.58     foreign      Foreign
```

This illustrates what happens when there is an error collecting statistics for one of the groups defined by the `accrue by` *varlist*. Namely, an informative message is printed on the screen so that you might investigate further to assure yourself that the program is running correctly and the statistic is posted to the new dataset as a missing value for this case. The important fact is that the `accrue` command continues running.

## Example 2

In this example, we investigate the model that the log of wages is dependent on the age, race, and whether or not the person is a college graduate. What we would like to do is to collect the coefficient of the race of the person for each of the years in the data (1968–1988) and then look at how that coefficient changes over time.

```
. accrue query
by:      year
coeffs:  race
cnames:  coef
stderr:  race
snames:  stderr
using:   example
cmd:     reg ln_wage age race collgrad
. accrue exec replace
 output omitted
. use example, clear
(NLS Women 14-24 in 1968)
. list
```

```
          year        coef      stderr
  1.        68  -.06686963    .0171553
  2.        69  -.06072424   .01765563
  3.        70   -.0406756   .01329873
  4.        71  -.07260486   .01362239
  5.        72  -.05767036   .01629522
  6.        73  -.07416148   .01447637
  7.        75  -.02900111   .01583388
  8.        77  -.04672074   .01626917
  9.        78   -.0587908   .01671812
 10.        80  -.06534773   .01792613
 11.        82  -.07970412   .01846965
 12.        83  -.05474226   .02097439
 13.        85  -.06510443   .02156693
 14.        87  -.06886119   .02220777
 15.        88  -.12281426   .02345575
```

## Example 3

In this example, I highlight the use of `accrue` to perform an ado-command across a `by` *varlist*. The standard Stata syntax specifies that commands may be prefixed with `by` *varlist*`:` in order to execute the command for each group defined by the `by` *varlist*. However, this does not work if the command is an ado-command. The reason for this is that the information associated with the `by` *varlist* is not available to the author of the ado-command. To circumvent this shortcoming, many ado-file authors include a `by()` option in their command to allow the execution of the command across a `by` *varlist*. Not all of the ado-files include this option and some that have this option use the `by` *varlist* for other meanings (such as to specify strata in the epitab commands). Using the `accrue` commands will allow us to run any command across a `by` *varlist*.

We once again return to the automobile dataset and wish to execute the `hlogit` command for each of the groups defined by the `rep78` variable. We want to model `foreign` by the `mpg`, `price`, and `weight` variables. We are not interested in collecting statistics for each of the models that we collect—just in running each of the models. In this example, there are only 5 groups so we could just have easily run each of the models ourselves, but there may be times when you have hundreds of groups.

```
. accrue query
by:      rep78
cmd:     hlogit foreign mpg price weight

. accrue exec noisily
proceeding, but no statistics specified to accrue

rep78=1
outcome does not vary.

rep78=2
outcome does not vary.

rep78=3
Logit Regression with Huber standard errors           Number of obs =      30
Log Likelihood =         0                            Pseudo R2     =  1.0000
------------------------------------------------------------------------------
 foreign |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |  -10.95414          .          .       .               .         .
   price |   .0097869   .0002231     43.865   0.000       .0093496    .0102242
  weight |  -.1462182          .          .       .               .         .
   _cons |   532.9895          .          .       .               .         .
------------------------------------------------------------------------------

rep78=4
Logit Regression with Huber standard errors           Number of obs =      18
Log Likelihood = -3.6071859                           Pseudo R2     =  0.7109
------------------------------------------------------------------------------
 foreign |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |  -.0652006   .2750039     -0.237   0.813      -.6041984    .4737973
   price |   .0015947   .0004208      3.790   0.000         .00077    .0024194
  weight |  -.0052838    .001749     -3.021   0.003      -.0087117   -.001856
   _cons |      6.348   8.661827      0.733   0.464      -10.62887    23.32487
------------------------------------------------------------------------------

rep78=5
Logit Regression with Huber standard errors           Number of obs =      11
Log Likelihood = -1.388e-07                           Pseudo R2     =  1.0000
```

```
------------------------------------------------------------------------------
 foreign |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |   13.93248          .        .       .              .           .
   price |   .0455436   .0255462      1.783   0.075      -.004526    .0956132
  weight |   .3942702   .0476808      8.269   0.000       .3008174    .4877229
   _cons |  -1451.998          .        .       .              .           .
------------------------------------------------------------------------------
```

## References

Gould, W. W. 1995. dm27: An improved collapse, with weights. *Stata Technical Bulletin* 24: 40–43.

Royston, P. 1995. ip9: Repeat Stata command by variable(s). *Stata Technical Bulletin* 27: 3–5.

## dt3 Reading EpiInfo datasets into Stata

R. Mark Esman, Stata Corporation, FAX 409-696-4601, stata@stata.com

`epi2dct` is a stand-alone DOS executable program written to convert data produced with EpiInfo into a format readable by Stata. `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, the source code is included.

I would appreciate feedback from those of you that use EpiInfo as a matter of course. For the few examples that I could try, the program always worked, but I am sure there will be instances where all does not translate to a dictionary file without problems. In the future, I will maintain updates of this source code 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*

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 uses a subset of data from the `auto.dta` dataset that accompanies Stata.

```
13 1
 EXAMPLEFIL   24   1  30  57   1   0   0 112 Example file for EPI2DCT program.
 MAKEMODEL     1   3  30  16   3   1  18 112 Make and Model
 PRICE         1   4  30   7   4   0   5 112 Price
 MILEAGEMPG    1   5  30  15   5   0   5 112 Mileage (mpg)
 REPAIRRECO    1   6  30  20   6   0   5 112 Repair Record 1978
 HEADROOMIN    1   7  30  16   7 101   4 112 Headroom (in.)
 TRUNKSPACE    1   8  30  23   8   0   5 112 Trunk space (cu. ft.)
 WEIGHTLBS     1   9  30  15   9   0   5 112 Weight (lbs.)
 LENGTHIN      1  10  30  14  10   0   5 112 Length (in.)
 TURNCIRCLE    1  11  30  19  11   0   5 112 Turn Circle (ft.)
 DISPLACEME    1  12  30  24  12   0   5 112 Displacement (cu. in.)
 GEARRATIO     1  13  30  12  13 102   5 112 Gear Ratio
 CARTYPE       1  14  30  10  14   0   1 112 Car Type
AMC Concord        4099   22    3 2.5   11 2930  186   40  121 3.580!
AMC Pacer          4749   17    3 3.0   11 3350  173   40  258 2.530!
AMC Spirit         3799   22      3.0   12 2640  168   35  121 3.080!
Buick Century      4816   20    3 4.5   16 3250  196   40  196 2.930!
Buick Electra      7827   15    4 4.0   20 4080  222   43  350 2.410!
Buick LeSabre      5788   18    3 4.0   21 3670  218   43  231 2.730!
Buick Opel         4453   26      3.0   10 2230  170   34  304 2.870!
Buick Regal        5189   20    3 2.0   16 3280  200   42  196 2.930!
Buick Riviera     10372   16    3 3.5   17 3880  207   43  231 2.930!
Ford Fiesta        4389   28    4 1.5    9 1800  147   33   98 3.150!
Audi 5000          9690   17    5 3.0   15 2830  189   37  131 3.201!
BMW 320i           9735   25    4 2.5   12 2650  177   34  121 3.641!
Honda Accord       5799   25    5 3.0   10 2240  172   36  107 3.051!
Mazda GLC          3995   30    4 3.5   11 1980  154   33   86 3.731!
Toyota Celica      5899   18    5 2.5   14 2410  174   36  134 3.061!
```

which we can convert to a Stata dictionary file using

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

The created dictionary file will be written as

```
dictionary {
        str18 makemode %18s "Make and Model"
        _column(19)
        price %5f "Price"
        mileagem %5f "Mileage (mpg)"
        repairre %5f "Repair Record 1978"
        headroom %4f "Headroom (in.)"
        _column(38)
        trunkspa %5f "Trunk space (cu. ft.)"
        weightlb %5f "Weight (lbs.)"
        lengthin %5f "Length (in.)"
        turncirc %5f "Turn Circle (ft.)"
        displace %5f "Displacement (cu. in.)"
        gearrati %5f "Gear Ratio"
        _column(68)
        cartype %1f "Car Type"
}
AMC Concord        4099   22    3 2.5   11 2930  186   40  121 3.580!
AMC Pacer          4749   17    3 3.0   11 3350  173   40  258 2.530!
AMC Spirit         3799   22      3.0   12 2640  168   35  121 3.080!
Buick Century      4816   20    3 4.5   16 3250  196   40  196 2.930!
Buick Electra      7827   15    4 4.0   20 4080  222   43  350 2.410!
Buick LeSabre      5788   18    3 4.0   21 3670  218   43  231 2.730!
Buick Opel         4453   26      3.0   10 2230  170   34  304 2.870!
Buick Regal        5189   20    3 2.0   16 3280  200   42  196 2.930!
Buick Riviera     10372   16    3 3.5   17 3880  207   43  231 2.930!
Ford Fiesta        4389   28    4 1.5    9 1800  147   33   98 3.150!
Audi 5000          9690   17    5 3.0   15 2830  189   37  131 3.201!
BMW 320i           9735   25    4 2.5   12 2650  177   34  121 3.641!
Honda Accord       5799   25    5 3.0   10 2240  172   36  107 3.051!
Mazda GLC          3995   30    4 3.5   11 1980  154   33   86 3.731!
Toyota Celica      5899   18    5 2.5   14 2410  174   36  134 3.061!
```

which can easily be read into Stata using the `infile` command.

```
. infile using auto.dct
```

## os16 Importing Stata graphs into word processors on the Macintosh

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

The question of how best to import Stata's graphs into a word processor comes up often in email and technical calls. Although the solution is quite simple, the difficulty lies in getting publication-quality results. As I've worked with most all desktop publishing (DTP) packages on the Macintosh and also have intimate knowledge of Stata for the Macintosh's internal workings, I can give some insight on how to get the best results from Stata's graphs. Many of the suggestions, limitations, and improvements apply to Stata for Windows as well, so this article may be of some interest to Windows users. In a future version of Stata, we will correct some of the limitations imposed by the current Stata and the Macintosh.

You can import Stata's graphs into your favorite word processor, or any DTP package for that matter, using one of two methods. The easiest method is to bring up the graph in Stata and select **Copy** from the **Edit** menu (make sure the Graph window is the frontmost window). Then switch to the other application and select **Paste** from the **Edit** menu. Unfortunately, this doesn't always give the best results. This is due to a limitation of the Macintosh, what some consider a limitation of Stata, and also a limitation of some applications. The Macintosh's limitation is that it uses an integer-based coordinate system for its graphics. This is adequate for most uses, but can produce poor results in certain situations. To see how this can affect an application, create a pie chart in Excel and paste it into Word. Scale the image to a small size and you will see that the lines that make up the slices no longer stop at the edge of the circle.

This limitation affects Stata because Stata draws its own fonts using lines and arcs. The Macintosh can't handle the level of detail required in drawing the text (and you also can't directly edit it as text). The text in Stata's graphs comes out poorly due to rounding error and this creates bitmap-like output. Stata tries to compensate for this by scaling the drawing area to a larger size and outputting the graph at 300 dpi. However, many DTP applications will only maintain the standard 72 dpi when importing a graphic.

WordPerfect initially works the same as most other applications in that if you paste a Stata graph into it and then print it, you will get the bitmap-like text described above. The trick is to have WordPerfect convert the graph from the Mac's internal PICT2 format to its own internal graphic format. You can do this by double-clicking on the graph. This will bring up a graphics editor in which you may edit the graphic or even add your own text. Although the text output is improved, it's not perfect as you can still see some small gaps in the fonts. Both Word and Nisus Writer have graphic editors, but neither one make any improvements to Stata's text.

Another possible limitation is how some applications treat Stata's circles. On the Macintosh (and in Windows), there are two ways of creating circles. One method is by drawing a 360° arc in a square bounding box. The other is to draw an oval in a square bounding box. Stata uses the former technique which can to lead to strange results in some applications when they improperly convert the image into their own format. During my test with Canvas, the circles didn't even show up because it misinterpreted the circle as an arc that had the same starting and ending point. Some applications printed the circles just fine but rescaling them led to unpredictable results. One application even internally represented Stata's circles as three joined arcs. The table below shows the result of pasting a graph into several common applications.

| | Text OK | True circles | Objects ungrouped | Preserve line thickness |
|---|---|---|---|---|
| Word 6.0 | no | no | n/a | no |
| WordPerfect 3.5 | yes* | yes | n/a | no |
| Nisus Writer 4.06 | no | yes | n/a | no |
| Adobe PageMaker*** | no | n/a | n/a | no |
| QuarkXPress 3.31*** | no | n/a | n/a | no |
| Freehand 5.5 | yes | yes | yes | yes |
| Illustrator 5.5** | n/a | n/a | n/a | n/a |
| Canvas 3.5.3 | no | yes | yes | yes |

| | |
|---|---|
| * | after using graphic editor |
| ** | I could not get Illustrator to paste a Stata graph from the clipboard |
| *** | Neither PageMaker or QuarkXPress allow editing of images |
| *Text OK* | does the text appear as intended? |
| *True circles* | can the graph's circles be edited as true circles? |
| *Objects ungrouped* | this is a matter of taste and only applies to illustration packages |
| *Preserve line thickness* | do the resulting line thicknesses match the intended line thicknesses? |

You can avoid these problems by using the other method of importing Stata graphs—creating an encapsulated postscript (EPS) file. To create an EPS file from a Stata graph, make sure you have the LaserWriter printer driver (version 8.3.3 is the latest) installed and selected from the Chooser. Then, bring up your graph in Stata and select **Print Graph...** from the **File** menu. You will be presented with the following print dialog.

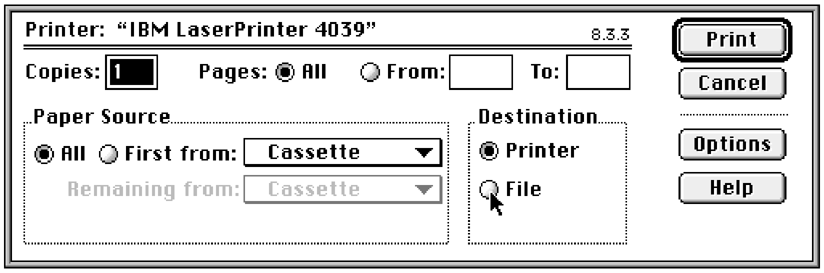
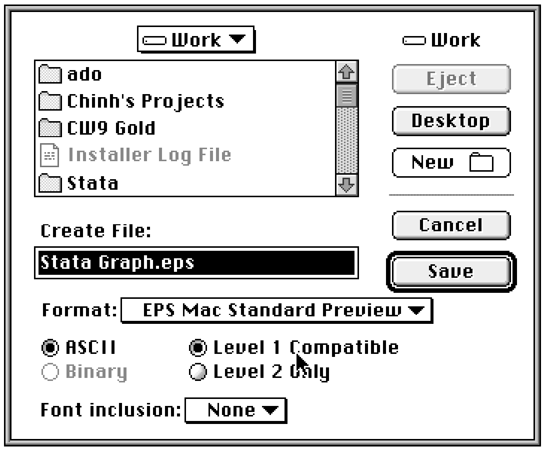

Printer: "IBM LaserPrinter 4039" 8.3.3
Copies: 1 Pages: [✓ All] From: To:
Paper Source: [✓ All] First from: Cassette
Remaining from: Cassette
Destination: [✓ Printer] File

Figure 1. The Apple LaserWriter print dialog

Select the **File** radio button in the Destination group box (the **Print** button will then change to **Save**) and then click on **Save** or press *Return*. You will then be presented with the following **Save** dialog.



Create File: Stata Graph.eps

Format: EPS Mac Standard Preview

[✓ ASCII] Binary

[✓ Level 1 Compatible] Level 2 Only

Font inclusion: None

Figure 2. The EPS save file dialog

Leave all the options as is but change the **Format** popup to "EPS Mac Standard Preview" so that you can see a preview of the image when placed in your DTP application. If you are going to use this EPS file on another platform such as Windows, then change the **Format** popup to EPS No Preview. If what you're seeing is different from what's above, you either do not have the LaserWriter driver selected from the Chooser, or you do not have a recent enough version of the driver (try Apple's homepage at http://www.apple.com).

The advantage of creating an EPS file is that all the top DTP packages and word processors can import them and they can be easily used on other platforms. Another advantage EPS files have over pasting from the clipboard is that EPS files are not limited by complexity and size. Many packages, especially Word, would not paste fairly complex graphs or graphs with a large number of points because they could not handle the number of objects. The disadvantages are that no package I know of supports editing of EPS files and EPS files can become quite large.

It's difficult to make a recommendation on which DTP application is best with Stata. I would recommend always importing your graphs as EPS files if you're not concerned with editing them before publication. If you do have to edit the graphs but don't have FreeHand available, I'd recommend using any of the illustration packages, WordPerfect or Nisus Writer and redoing all of the text in the graphs.

## sed10 Patterns of missing data

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

In many circumstances, the pattern of missing data is of great interest (and is used by Stata's `impute` command for filling in missing values). The `pattern` command can help find variables that may be particularly problematic (but also see the `mv` option to `codebook`), and it can be used to look for actual missing value patterns and to compare such patterns across groups (e.g., treatment and control groups, males and females, etc.).

The syntax of `pattern` is

`pattern` *varlist* [ `if` *exp* ] [ `in` *range* ]

`pattern` gives a simple description, and count, of the number of patterns and what they look like. A maximum of 55 variables can be included (though I usually use many fewer than that) so that the entire pattern fits on the screen. The only allowed options are `if` and `in`. The variables in the *varlist* can be either string or numeric.

An example using the Stata-supplied auto data and three of its variables shows what the output looks like

```
. pattern price mpg rep78
     COUNT    PCT    PATTERN
  1.     5   6.76        XX.
  2.    69  93.24        XXX
Total: 74
```

There are four columns of information and a total at the bottom. The first column gives the number of patterns of missing data that are found (note that observations with no missing data, if any, form the last pattern shown). If there are more than just a few patterns, having this enumeration can be useful when discussing, or writing about, the different patterns. The second column, headed `COUNT`, shows the number of observations that have the displayed pattern. The third column (`PCT`) shows what percentage of the total number of observations have the displayed pattern. The final column shows the pattern. Each column of the pattern refers to one particular variable (shown in the order listed in the `pattern` command). Each row of each column contains either a dot (".") or an "`X`". A dot means the variable is missing and an `X` means it is not missing. Thus, in the example above, there are only two patterns for the 74 observations; in one pattern, the first, five observations (6.76% of the total) are missing for the final variable used (here `rep78`); in the second pattern, no variables have missing values for 69 of the 74 observations. At the bottom of the display, following the word `Total` is the total number of observations examined. Other examples are shown in the help file.

| sg51 | Inference about correlations using the Fisher z-transform |
|---|---|

John R. Gleason, Syracuse University, 73241.717@compuserve.com

Suppose $(x_1, y_1), \ldots, (x_n, y_n)$ is a bivariate random sample of size $n$ where $X$ and $Y$ are random variables with product-moment correlation coefficient $\rho$. That is,

$$\rho = \frac{E[(X - E(X))(Y - E(Y))]}{\sigma_X \sigma_Y}$$

where $\sigma_X$ and $\sigma_Y$ are the standard deviations of $X$ and $Y$. Let $r$ denote the usual sample correlation coefficient,

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y}$$

where $\bar{x}$, $\bar{y}$, $s_x$, and $s_y$ are sample means and standard deviations calculated from the $n$ pairs $(x_i, y_i)$. If the joint distribution of $X$ and $Y$ is bivariate normal, the statistic $t = r\sqrt{(n-2)/(1-r^2)}$ is distributed as Student's $t(n-2)$ when $H_0$: $\rho = 0$ is true. The resulting $t$ test is widely used—many textbooks describe no other form of statistical inference for correlation coefficients. (The `sig` option of Stata's `pwcorr` command reports $p$-values for this test.)

On the other hand, the distribution of

$$z = T(r) \equiv \frac{1}{2}\log\left(\frac{1+r}{1-r}\right)$$

is approximately $N(T(\rho), (n-3)^{-1})$ if the joint distribution of $X$ and $Y$ is not too far from bivariate normal and $n$ is not too small. $T(r)$ is the famous *Fisher $z$-transform* of the correlation coefficient, whose inverse is given by

$$r = T^{-1}(z) \equiv \frac{e^{2z} - 1}{e^{2z} + 1}$$

Because $T(r)$ is approximately Gaussian with variance that is known and independent of $\rho$, a very simple approach to (approximate) inference is possible: perform the inference in terms of $z = T(r)$, and then transform back to correlation format using $T^{-1}(z)$. Unlike the $t$ test described earlier, this approach permits tests of nonzero values of $\rho$, as well as the construction of confidence intervals.

This insert presents the first installment of a suite of commands for statistical inference about correlation coefficients via the Fisher $z$-transform $T(r)$. These initial commands permit the user to form a correlation matrix, construct confidence intervals for elements of the matrix, and graph sets of correlations and their confidence limits. A second installment will present commands for comparing pairs of correlation coefficients in various settings.

## Confidence intervals for correlation coefficients

The command `z_r` creates and stores a matrix of correlation coefficients and a matrix of confidence limits for each nontrivial correlation coefficient. The syntax of `z_r` is

**`z_r`** *varlist* [ *weight* ] [ **`if`** *exp* ] [ **`in`** *range* ] [ **`,`** **`level(`***#***`)`** **`nocorr`** **`nolimit`** ]

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

`level(`#`)` sets the individual confidence level; by default, 95% confidence intervals are formed.

`nocorr` suppresses the display of the matrix of correlation coefficients.

`nolimit` suppresses the display of the matrix of confidence limits.

To illustrate using the familiar `auto.dta`, the example below calculates some correlations and confidence limits using the data for foreign-made automobiles:

```
. z_r price mpg weight length displ if foreign
(sample correlations, n=22)
          price      mpg   weight   length    displ
 price   1.0000
   mpg  -0.6313   1.0000
weight   0.8855  -0.6829   1.0000
length   0.8195  -0.6482   0.9106   1.0000
 displ   0.8114  -0.7498   0.9507   0.8809   1.0000

(lower\upper 95% confidence limits)
          price      mpg   weight   length    displ
 price   1.0000  -0.2858   0.9518   0.9224   0.9187
   mpg  -0.8316   1.0000  -0.3669  -0.3118  -0.4798
weight   0.7403  -0.8576   1.0000   0.9626   0.9797
length   0.6079  -0.8402   0.7936   1.0000   0.9498
 displ   0.5925  -0.8900   0.8831   0.7306   1.0000
```

The first matrix displayed is a $5 \times 5$ matrix of sample correlation coefficients $r$, each based on $n = 22$ $x$-$y$ pairs. The second matrix shows the lower (below the diagonal) and upper (above the diagonal) 95% confidence limits for each correlation in the first matrix. Thus, the 95% confidence limits for the correlation between **weight** and **displ** are $T^{-1}(T(.9507) - 1.96/\sqrt{19}) = .8831$, and $T^{-1}(T(.9507) + 1.96/\sqrt{19}) = .9797$.

If, as in the preceding example, the number of variables in the *varlist* is rather small, the output of **z_r** is easy to view. Otherwise, the correlation and confidence limit matrices may be too large to readily digest, and a different display style may be preferred. Consider the command

```
. z_r price-foreign, nocorr nolimit level(.975)
```

which calculates $r$ and its 97.5% confidence limits for each of the 55 pairs of variables in the *varlist* **price-foreign**, using data from $n = 69$ automobiles. The **nocorr** and **nolimit** options suppress the display of the resulting pair of $11 \times 11$ matrices. The correlation coefficients and their confidence limits are nevertheless stored, in the matrices **S_1** and **S_2**, respectively. (See *Saved Results*, below, for additional details about what is saved.)

The command **z_rci** then displays subsets of matrices **S_1** and **S_2**. Its syntax is

**z_rci** *varlist*

where *varlist* is any subset of the variables represented in the matrix **S_1**, specified in any order. For example,

```
. z_rci length hdroom weight displ mpg
Variables                r      97.5% confidence limits
length, hdroom        0.5240      0.2967     0.6951
length, weight        0.9478      0.9111     0.9696
length, displ         0.8621      0.7721     0.9182
length, mpg          -0.8037     -0.8820    -0.6821
hdroom, weight        0.4795      0.2415     0.6630
hdroom, displ         0.4763      0.2377     0.6607
hdroom, mpg          -0.3996     -0.6038    -0.1462
weight, displ         0.9316      0.8842     0.9600
weight, mpg          -0.8055     -0.8832    -0.6849
displ, mpg           -0.7434     -0.8437    -0.5929
```

Finally, note that Fisher $z$ confidence limits depend on the sample only through $n$ and $r$, and thus they can be calculated without access to the raw data on which $r$ is based. (But see *Remarks*, below, for a warning about the wisdom of this practice.) For this reason, there is an immediate command for constructing confidence intervals for correlation coefficients. The syntax is

**z_rcii** *n r* [ , **l**evel(*#*) ]

where *r* is a sample correlation coefficient, *n* is the sample size on which it is based, and the **level** option is as described for **z_r**. To illustrate, given $r = .5854$ based on $n = 50$ cases, a 90% confidence interval is obtained as

```
. z_rcii 50 .5854, l(90)
      n        r      90% confidence limits
     50     0.5854      0.4059     0.7214
```

## Graphing correlation coefficients and their confidence intervals

The command `z_rplt` is a simple tool for visualizing correlation coefficients and their confidence intervals, using a version of Cleveland's (1985) dot chart. Specifically, represent the possible range of a correlation as a linear array of 20 dots, placed at the values $r = \pm 0.1, \pm 0.2, \ldots, \pm 1$, with the midpoint $r = 0$ marked by another symbol (say, $+$). Then, represent any given sample correlation by a plot symbol positioned appropriately along the axis implied by the series of dots, and the associated confidence interval by a line extending from its lower to upper limit. A text version of such a dot chart might look like

```
. . . . . . . . . . + . . .-----O---- . .
```

From this display, one can extract the values of *r* and its confidence limits to about one decimal place accuracy, which is often as much accuracy as the data warrant. (*Very* large sample sizes are required to estimate a correlation coefficient to two decimal place accuracy.)

The command `z_rplt` draws plots whose elements are graphical versions of the above dot chart, using the correlations and confidence limits stored in matrices `S_1` and `S_2` by `z_r`. Its syntax is

`z_rplt` *rvarlist* [ , `column(`*cvarlist*`)` `pen(`*penspec*`)` `symbol(`*symspec*`)` *graph_options* ]

where each of *rvarlist* and *cvarlist* is any subset of the variables represented in matrix `S_1`, specified in any order. *graph_options* includes most of the options allowed with `graph, twoway`; see *Remarks* below for details. The remaining options are explained in the examples that follow.

`z_rplt` can produce two styles of plots. By default, it produces a triangular display in which each element is a dot chart portraying *r* and its confidence limits for a pair of the variables in *rvarlist*. That is, the default style is a graphical rendition of the lower half of a correlation matrix. To illustrate, *r* and confidence limits are plotted for the same five variables used to demonstrate `z_rci` above:

```
. z_rplt length hdroom weight displ mpg
```

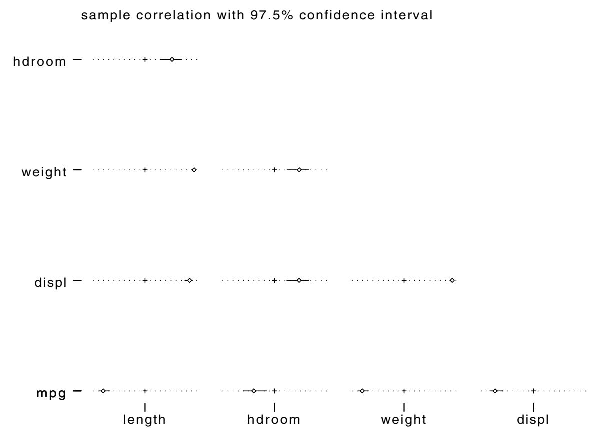


Figure 1. The Default Style of `z_rplt`.

As the number of variables in *rvarlist* increases, the default triangular display style of `z_rplt` becomes more inefficient: each dot chart is rendered as a smaller image and more display space goes unused. As an alternative, the `column(`*cvarlist*`)` option presents a dot chart for the correlation of each variable in *rvarlist* (as rows) with each variable in *cvarlist* (as columns). That is, the `column` option draws dot charts for a rectangular submatrix of the correlation matrix stored by `z_r`. This style uses display space more efficiently and makes it easy to focus attention on correlations involving a particular subset of variables—namely, the elements of *cvarlist*.

For example, suppose that in `auto.dta`, we wish to focus on correlations with the variables `foreign` and `price`:

```
. z_rplt grat mpg hdr trunk len displ weight turn, col(for price) t1("Automobile Data")
```

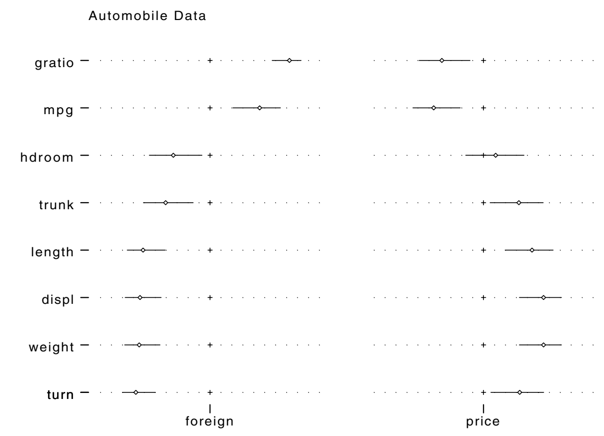


Figure 2. The Rectangular Style of `z_prly`

For either style of dot chart plot, the `symbol(`*symspec*`)` and `pen(`*penspec*`)` options may be supplied. *symspec* is a single character that specifies the plot symbol for the sample $r$, any character in the string "`OTSopd.i`"; see [3] symbol. The default symbol is `d`, a small diamond. *penspec* must be a string of integers that specify pens for Stata's `graph` command; see [3] pens. This provides a way of controlling the color (on color output devices) and thickness (in printed output) of dot chart elements. Specifically, `pen(`*abc*`)` assigns pen $a$ to the sample correlation $r$, pen $b$ to the line representing the confidence interval, and pen $c$ to the dots that form the pseudo-axis of the dot chart; the default is `pen(231)`. Often, it will be useful to choose $c$ to obtain faint rather than bold dots in printed output. Also note that setting $b = 0$ will erase the axis between the confidence limits so that the confidence interval is rendered as a gap in the axis. This may be an effective presentation when the axis is short and the dots are dense.

## Saved Results

The command `z_r` saves a correlation matrix in the matrix `S_1`, and its associated confidence limits in the matrix `S_2`. Further, `z_r` creates a $1 \times 3$ matrix `S_3` whose elements are the number of observations $n$ used to compute each $r$, the dimension $p$ of the correlation matrix `S_1`, and the level of the confidence intervals stored in the matrix `S_2`. Thus, with the automobile data, consider the following command:

```
. z_r price mpg weight length displ if !foreign
(sample correlations, n=52)
          price      mpg   weight   length    displ
 price   1.0000
   mpg  -0.5043   1.0000
weight   0.6724  -0.8759   1.0000
length   0.5009  -0.8543   0.9210   1.0000
 displ   0.6881  -0.7473   0.8439   0.7586   1.0000
(lower\upper 95% confidence limits)
          price      mpg   weight   length    displ
 price   1.0000  -0.2683   0.7987   0.6808   0.8091
   mpg  -0.6832   1.0000  -0.7925  -0.7582  -0.5960
weight   0.4893  -0.9272   1.0000   0.9541   0.9078
length   0.2641  -0.9141   0.8657   1.0000   0.8546
 displ   0.5112  -0.8474   0.7419   0.6125   1.0000
```

The correlations and confidence limits in this display are the elements of the matrices `S_1` and `S_2`, respectively. The matrix `S_3` takes this form:

```
. matrix list S_3
S_3[1,3]
         n      p  level
z_r     52      5    .95
```

The command `z_r` also saves in the `S_#` macros:

| | |
|---|---|
| `S_1` | the command name, `z_r` |
| `S_2` | the [ `if` ] [ `in` ] clause |

So, continuing the last example, the macros `S_1` and `S_2` would have these contents:

```
. di "$S_1" _newline "$S_2"
z_r
if !foreign
```

The command `z_rplt` uses the matrix `S_0` for temporary storage, but erases it before exiting. There are no other saved results.

### Utilities

It may sometimes be useful to copy the matrices saved by `z_r` to other locations to protect them from destruction by `z_r` or some other command. For example, in the automobile data, it might be interesting to compare correlations among a set of variables for foreign autos with their counterparts for domestic autos. We might then wish to avoid recomputing correlation matrices each time we switch attention from one type of auto to the other.

The command `z_rcopy` copies matrices saved by `z_r` to or from the matrices `S_1`, `S_2`, and `S_3`. Its syntax is

`z_rcopy` , { `from`(*m1 m2 m3*) | `to`(*m1 m2 m3*) }

where *m1*, *m2*, and *m3* are the names of three matrices. Exactly one of the options `to()` and `from()` must be present. If `to()` is present, `z_rcopy` copies the matrix `S_1` to the matrix *m1*, matrix `S_2` to matrix *m2*, and matrix `S_3` to matrix *m3*. If the `from()` option is present, `z_rcopy` copies the three matrices in the reverse direction.

To illustrate,

```
. z_r price mpg weight length displ if !foreign, nocor nolim
. z_rcopy, to(dom1 dom2 dom3)
```

computes a 5 × 5 correlation matrix and associated confidence limits using data for domestic autos, and then copies the matrices `S_1`, `S_2`, and `S_3` to matrices `dom1`, `dom2`, and `dom3`, respectively. The corresponding matrices for foreign autos can then be created by calling `z_r` again. But, note that it may be convenient to collect sets of three matrix names into global macros. For example, continuing the last example,

```
. global foreign "for1 for2 for3"
. z_r price mpg weight length displ if foreign, nocor nolim
. z_rcopy, to($foreign)
```

computes a 5 × 5 correlation matrix and associated confidence limits using data for foreign autos, and then copies the results to the matrices named in the macro `foreign`.

Commands such as `z_rci` and `z_rplt` of course operate on the current contents of the matrices `S_1`, `S_2`, and `S_3`, and `z_rcopy` may be used to fill those matrices with results computed earlier. Continuing our example,

```
. global domestic "dom1 dom2 dom3"
. z_rcopy, from($domestic)
. z_rplt price weight displ mpg
```

draws a dot chart for correlations computed earlier using observations on domestic autos.

Clearly, it is essential that the triplet of matrices stored by `z_r`, and assumed by commands such as `z_rplt`, be moved about as a unit. Collecting matrix names in macros, as described above, is a simple way of packaging output from the command `z_r`. Nevertheless, commands such as `z_rplt` must verify the contents of the matrices `S_1`, `S_2`, and `S_3` before proceeding. This is done with the command `z_rvrfy`, whose syntax is

`z_rvrfy` *m1 m2 m3*

where *m1*, *m2*, and *m3* are three matrix names. `z_rvrfy` checks to see that its arguments *appear* to be three matrices saved by a single invocation of the command `z_r`. (It can, however, be fooled!) Various commands in the `z_r` suite invoke `z_rvrfy`, but it is available to the user as well. To illustrate in our running example,

```
. z_rvrfy dom1 dom2 dom3
. z_rvrfy dom1 dom2 for3
output from z_r not found
r(499);
```

### Remarks

1. `z_r` handles missing data by performing *casewise deletion*, the same approach used by Stata's `correlate` command.

2. `z_r` and `z_rci`, typed without arguments, display the entire correlation matrix `S_1` and the entire confidence limit matrix `S_2` — after checking their contents with `z_rvrfy`. Thus, the commands `z_r` and `z_rci` display exactly the same numerical results, but in different styles.

3. The argument of the `level()` option of the `z_r` and `z_rcii` commands may be given either as a fraction or as a percentage. For example, `level(99.5)` and `level(.995)` both request 99.5% confidence intervals.

4. The following options of the `graph` command are set by `z_rplt` and may not be altered by the user: `connect`, `noaxis`, `xlabel`, `ylabel`, `yreverse`. Other graph options (especially `saving`) may be supplied.

5. Distributional properties of Fisher's $z$-transform are derived under the assumption that the sample is drawn from a bivariate normal distribution. While mild non-normality will likely have little impact on the behavior of $T(r)$, certain stronger forms of non-normality are known to seriously distort inferences based on $T(r)$. Also note that the $t$ test of $H_0$: $\rho = 0$ is (at least approximately) valid under broader conditions than bivariate normality, although this is of little relevance if 0 is not the right value of $\rho$ to entertain.

6. The sample correlation $r$ is notoriously sensitive to outliers, and so immediate commands (such as `z_rcii`) should be used with caution: If the raw data are unavailable, there is no way to do diagnostic checks, regardless of how one uses $r$ to perform inference about $\rho$.

7. Bootstrapping is an effective way to robustify inference about correlations, and the $z$-transform, $T(r)$, plays an important role in this approach. See Efron and Tibshirani (1993) for many details.

## References

Cleveland, W. S. 1985. *The Elements of Graphing Data.* Monterey, CA: Wadsworth.

Efron, B. and R. J. Tibshirani 1993. *An Introduction to the Bootstrap.* London: Chapman & Hall.

Fisher, R. A. 1921. On the probable error of a coefficient of correlation deduced from a small sample. *Metron* 1(4): 3–32.

## sg52 Testing dependent correlation coefficients

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

When one wants to test whether two *dependent* correlation coefficients are equal, there is little software available for help (actually there is little literature available). Yet often one wants to test whether the correlation of $Y$ with $X_1$ is equal to the correlation of $Y$ with $X_2$ (where the $Y$'s are the same). The program `corcor` (and its immediate version `corcori`) do this. Further, these programs call `fisher.ado` and `fisheri.ado` which implement Fisher's $z$-transformation for sample correlation coefficients and clearly can be used with `corcor` (stands for correlated correlations).

The only other software that I know of that performs this test is Dallal's STAT-SAK (Dallal 1992, 3). The version in `corcor` is asymptotically equivalent to Dallal's version (Meng, Rosenthal and Rubin 1992; Dallal relies on Steiger 1980). Comparison of the results below with those from STAT-SAK, show great similarities.

The syntax for `corcor` and `corcori` is

`corcor` *var1 var2 var3* [ `if` *exp* ] [ `in` *range* ]

`corcori` *corr1 corr2 corr3 N*

`corcor` tests whether the correlation between *var1* and *var2* equals the correlation between *var1* and *var3*. The common variable must be entered first; the other two may be entered in any order. The immediate version asks for the correlation between *var1* and *var2*, the correlation between *var1* and *var3*, the correlation between *var2* and *var3*, and finally the sample size *N*.

The non-immediate version calls `fisher.ado`; the immediate version calls `fisheri.ado`. These ado-files use the Fisher $z$-transformation for the correlation coefficients; see `fisher.hlp`.

### Example

```
. corcori .63 -.03 -.19 15
Test Statistic (Z) =  1.6794   p-value:  0.0465   2-tailed p-value:  0.0931
```

For these same correlations, STAT-SAK gives a $z$ of 1.703 and a two-tailed $p$-value of .0885. This example is from Meng, Rosenthal and Rubin (1992); they also provide formulae for testing the heterogeneity of a set $(> 2)$ or correlated correlations, but this has not been implemented here.

## References

Dallal, G. 1992. an19: Stand-alone statistical tools. *Stata Technical Bulletin* 7: 3.

Meng, X.-L., R. Rosenthal, and D. B. Rubin. 1992. Comparing correlated correlation coefficients. *Psychological Bulletin* 111: 172–175.

Olkin, I. and J. Finn. 1990. Testing correlated correlations. *Psychological Bulletin* 108: 330–333.

Steiger, J. 1980. Tests for comparing elements of a correlation matrix. *Psychological Bulletin* 87: 245–251.

## sg53 Maximum-likelihood complementary log-log regression

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

A program designed to fit binary and proportional response data using complementary log-log (`cloglog`) regression in Stata first appeared as part of the author's Generalized Linear Models (GLM) program in January 1993. A revision to the `glm` command was made by Royston in May 1994. With Stata 4.0, `glm` is now a full Stata command. All of these versions incorporate the standard GLM method of using a scoring algorithm based on the expected second derivative of the log likelihood. This method is typically termed Fisher scoring. Each iteration of the scoring algorithm is a weighted least squares regression (IRLS) of an "adjusted dependent variable" on the predictive variables with a diagonal weight matrix. As a modification of the Newton–Raphson (N–R) algorithm, Fisher scoring has typically been chosen for fitting GLMs because of its robustness and its simplicity.

The essential difference between the full N–R algorithm and its Fisher scoring modification is that the former employs the observed rather than the expected information matrix. Both yield identical results when dealing with canonical linked GLMs; e.g., logistic regression. However, the differences in information become apparent when dealing with noncanonical models; e.g., `probit`, `cloglog`. Since estimated standard errors are derived from the information matrix, Fisher scoring results in slightly different standards errors than do models based on the full N–R algorithm. Fortunately, the differences are negligible unless employed on small data situations. Stata's `probit` command is estimated using N–R, hence the difference between its output and that of the `glm` command.

I have written a complementary loglog, or `cloglog`, ado program which is based on a N–R algorithm using an observed value information matrix. Moreover, I have constructed it such that it appears identical to other Stata maximum-likelihood output, e.g., Stata's `logit` and `probit` commands, which are both coded internally in Stata. In addition, I provide options which make it more versatile and useful in modeling than many other Stata `ml` commands.

### Syntax

`cloglog` *varlist* [*weight*] [`if` *exp*] [`in` *range*] [`, noconstant nolog group offset(`*offsetvar*`)`

`residual disp scale(`*scaleval*`) expec(`*#*`)` *maximize_options* ]

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

### Options

`noconstant` suppresses the estimation of a constant term.

`nolog` suppresses the display of the iteration log.

`group` specifies that the first variable in the *varlist* is the proportional numerator and the second variable in the *varlist* is the proportional denominator. The default is that there is only one group.

`offset(`*offsetvar*`)` specifies the variable that gives the offset of the values. Default is to use an offset of zero. See the documentation on the `glm` command for more information on the `offset`.

`residual` specifies that residuals should be calculated. If specified, the following variables will be created in your dataset:

| | |
|---|---|
| `_mu` | Fitted values |
| `_lp` | Linear predicted values |
| `_Pear` | Pearson chi-squared residuals |
| `_Dev` | Deviance based residuals |
| `_Hat` | Hat matrix diagonals |
| `_Anscom` | Anscombe residuals |
| `_StandP` | Standardized Pearson chi-squared residuals |
| `_StandD` | Standardized deviance residuals |

`disp` specifies to display the amount of binomial overdispersion.

`scale(`*scaleval*`)` allows you to change the scale on which the standard errors are calculated. See the documentation on the `glm` command for more information.

`expec(`*#*`)` specifies the number of iterations using Fisher scoring. The default is 2.

*maximize_options* control the maximization process; see [7] maximize in volume 1 of the Stata reference manuals.

### Example

For an example I shall use a dataset provided in the Stata 4.0 Manual referencing the `glm` command. As proportional data with $n$ as the denominator and `ldose` (the log dosage) as the explanatory variable, `glm` results are

```
. glm r ldose, f(bin n) l(c) nolog
Residual df   =         6                           No. of obs =         8
Pearson X2    =  3.294685                           Deviance   =  3.446418
Dispersion    =  .5491142                           Dispersion =   .574403
Binomial (N=n) distribution, cloglog link
---------------------------------------------------------------------------
       r |      Coef.   Std. Err.       z     P>|z|     [95% Conf. Interval]
---------+-----------------------------------------------------------------
   ldose |   22.04118   1.799365     12.249   0.000    18.51449    25.56787
   _cons |  -39.57232   3.240291    -12.213   0.000   -45.92318   -33.22147
---------------------------------------------------------------------------
```

Modeling the same data using cloglog results in

```
. cloglog r n ldose, g nolog
Cloglog Estimates                                  Number of obs =       8
                                                   Chi2(1)       =  280.76
                                                   Prob>Chi2     =  0.0000
Log Likelihood = -182.3425                         Pseudo R2     =  0.4350
---------------------------------------------------------------------------
       r |      Coef.   Std. Err.       z     P>|z|     [95% Conf. Interval]
---------+-----------------------------------------------------------------
   ldose |   22.04118   1.793089     12.292   0.000    18.52678    25.55557
   _cons |  -39.57232   3.229049    -12.255   0.000   -45.90114    -33.2435
---------------------------------------------------------------------------
```

Modeling the same data using SAS, which is also based on the full N–R method, yields these partial results:

```
             Analysis Of Parameter Estimates
Parameter    DF    Estimate     Std Err   ChiSquare  Pr>Chi
INTERCEPT     1    -39.5723      3.2290    150.1877  0.0000
LDOSE         1     22.0412      1.7931    151.1006  0.0000
SCALE         0      1.0000      0.0000           .       .
Log Likelihood   -182.3425
```

The square root of SAS $\chi^2$ values are identical to $z$ statistics as displayed in `cloglog` output; hence the results are consistent.

The comparative results show there to be little difference in the effect of the standard errors on the significance of the predictors into the model. However, this is not always the case. With respect to `cloglog` regression on small datasets, preference should be given to standard errors based on `cloglog` rather than on `glm`.

## sg54 Extended probit regression

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

The current Stata `probit` regression command limits modeling situations to binary response data. In order to model grouped data, one must use `bprobit`. No residual statistics can be directly produced using either commands. However, if neither residual analysis nor model checking are of concern, `probit` and `bprobit` may both be preferable to Stata's `glm` command, which also provides `probit` regression. The problem with the latter is that standard errors may be less accurate with small datasets. This situation is described in sg53: Maximum-likelihood complementary log-log regression; see above.

The new `eprobit` command allows both binary and proportional responses. Its syntax is

`eprobit` *varlist* [ `weight` ] [ `if` *exp* ] [ `in` *range* ] [, `noconstant nolog group offset(`*offsetval*`)`
`residual disp scale(`*scaleval*`) expec(#)` *maximize_options* ]

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

The options are as described in the previous article, sg53. Other options may be found in the accompanying help file.

I have found this program useful since I employ fit and residual analysis after all modeling tasks.

### Example

Using the auto dataset that comes with Stata, we shall model `foreign` on `mpg`, `length`, and `weight` using `eprobit` and Stata's `probit` and `glm` commands.

```
. glm foreign mpg length weight, f(bin) l(p) nolog
```

```
Residual df  =        70                               No. of obs =        74
Pearson X2   =  51.24021                               Deviance   =  53.68622
Dispersion   =   .732003                               Dispersion =   .766946

Bernoulli distribution, probit link
------------------------------------------------------------------------------
 foreign |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |  -.1040007    .054157     -1.920   0.055     -.2101465    .0021451
  length |  -.0015516   .0326583     -0.048   0.962     -.0655607    .0624576
  weight |   -.002292   .0010689     -2.144   0.032     -.0043871   -.0001969
   _cons |   8.437612   4.300635      1.962   0.050      .0085215     16.8667
------------------------------------------------------------------------------

. probit foreign mpg length weight, nolog

Probit Estimates                                       Number of obs =     74
                                                       chi2(3)       =  36.38
                                                       Prob > chi2   = 0.0000
Log Likelihood = -26.843111                            Pseudo R2     = 0.4039

------------------------------------------------------------------------------
 foreign |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |  -.1040006   .0515543     -2.017   0.044     -.2050451   -.0029561
  length |  -.0015517   .0333588     -0.047   0.963     -.0669337    .0638303
  weight |   -.002292   .0010912     -2.100   0.036     -.0044308   -.0001533
   _cons |   8.437624   4.327984      1.950   0.051     -.0450683    16.92032
------------------------------------------------------------------------------

. eprobit foreign mpg length weight, nolog resid

Probit Estimates                                      Number of obs =      74
                                                      Chi2(3)       =   36.38
                                                      Prob>Chi2     =  0.0000
Log Likelihood = -26.84311                            Pseudo R2     =  0.4039

------------------------------------------------------------------------------
 foreign |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |  -.1040006   .0515543     -2.017   0.044     -.2050451   -.0029561
  length |  -.0015517   .0333588     -0.047   0.963     -.0669336    .0638303
  weight |   -.002292   .0010912     -2.100   0.036     -.0044308   -.0001533
   _cons |   8.437624   4.327983      1.950   0.051     -.0450674    16.92032
------------------------------------------------------------------------------

Variables created:
 _mu (fit),                  _lp (linear predictor)
 _Pear (Pearson resid),      _Dev (deviance resid)
 _Anscom (Anscombe resid),   _Hat (hat matrix diag.)
 _StandP (standard Pearson), _StandD (standard deviance)
```

## sg55 Extensions to the brier command

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

The brier score decomposition command in Stata is very useful, but some extensions may make it more useful. `brier2` may be used in place of `brier` and provides the following additions to the output of `brier`: Mean probability of forecast is the mean of the probability; Correlation is the correlation between the judgments and the outcomes; Spiegelhalter's $z$-statistic is a standard normal test statistic for testing whether an individual Brier score is extreme; and ROC area is the same as the receiver operating curve produced by `lroc` (and by `ranksum2`) and the associated test is a test of whether the area is greater than 0.5.

The syntax of this new command is exactly as the original version of the `brier` command:

`brier2` *outcome forecast* [ `if` *exp* ] [ `in` *range* ] [ `, group(`*#*`)` ]

The only change is that the new `brier2` command provides additional output.

The option `group()` specifies the number of groups that will be used to compute the decomposition. `group(10)` is the default.

The `brier` data used below appears on the diskette as `brier.dta`. This, and the extensions, come from Redelmeier, Bloch, and Hickam (1991). A second ado-file, `briertst.ado`, is a test of the equality of two different Brier scores where an example is shown at the end of the listing.

```
. use brier
(J Clin Epi, 44:1141-6, 1991)
```

```
. brier2 y studa
Mean probability of forecast  0.4900   of outcome 0.5200
Correlation                   0.0974

Brier score                   0.2811
Spiegelhalter´s z-statistic   2.4366  p =         0.0074
ROC area                      0.5353  p(H0<=.5)=  0.3821
Sanders-modified Brier score  0.2896
Sanders resolution            0.1933
Outcome index variance        0.2496
Murphy resolution             0.0563
Reliability-in-the-small      0.0962
Forecast variance             0.0530
Excess forecast variance      0.0525
Minimum forecast variance     0.0005
Reliability-in-the-large      0.0009
2*Forecast-Outcome-Covar      0.0224

. brier2 y studb
Mean probability of forecast  0.5960   of outcome 0.5200
Correlation                   0.3114

Brier score                   0.2318
Spiegelhalter´s z-statistic   0.7243  p =         0.2344
ROC area                      0.6603  p(H0<=.5)=  0.0854
Sanders-modified Brier score  0.2362
Sanders resolution            0.1400
Outcome index variance        0.2496
Murphy resolution             0.1096
Reliability-in-the-small      0.0929
Forecast variance             0.0326
Excess forecast variance      0.0294
Minimum forecast variance     0.0032
Reliability-in-the-large      0.0058
2*Forecast-Outcome-Covar      0.0562

. brier2 y studc
Mean probability of forecast  0.6880   of outcome 0.5200
Correlation                   0.3643

Brier score                   0.2450
Spiegelhalter´s z-statistic   1.7276  p =         0.0420
ROC area                      0.7564  p(H0<=.5)=  0.0134
Sanders-modified Brier score  0.2468
Sanders resolution            0.1400
Outcome index variance        0.2496
Murphy resolution             0.1096
Reliability-in-the-small      0.1068
Forecast variance             0.0397
Excess forecast variance      0.0344
Minimum forecast variance     0.0053
Reliability-in-the-large      0.0282
2*Forecast-Outcome-Covar      0.0725

. brier2 y studd
Mean probability of forecast  0.6756   of outcome 0.5200
Correlation                   0.7047

Brier score                   0.1579
Spiegelhalter´s z-statistic   0.2124  p =         0.4159
ROC area                      0.8750  p(H0<=.5)=  0.0003
Sanders-modified Brier score  0.1565
Sanders resolution            0.0600
Outcome index variance        0.2496
Murphy resolution             0.1896
Reliability-in-the-small      0.0965
Forecast variance             0.0687
Excess forecast variance      0.0346
Minimum forecast variance     0.0341
Reliability-in-the-large      0.0242
2*Forecast-Outcome-Covar      0.1846

. briertst y studb studd
Comparison of two Brier scores     1.2926; p =  0.1961
```

## Reference

Redelmeier, D. A., D. A. Bloch, and D. H. Hickam. 1991. Assessing predictive accuracy: how to compare Brier scores. *Journal of Clinical Epidemiology* 44: 1141–1146.

# sg56 An ado-file implementation of a simplex-based maximization algorithm

Tavis Barr, Department of Economics, Columbia University

`simplex` is a likelihood-maximization routine based on the simplex method, also known as the Amoeba method, due to Nelder and Mead (1965). Its syntax is

`simplex` [ *varlist* ] [ `if` *exp* ] [`in` *range* ] `, from(`*matrix*`) lf(`*function*`)` [ `iterate(`*#*`) noconstant`

`step(`*#*`) span(`*#*`) tolerance(`*#*`) trace` ]

## Options

`from(`*matrix*`)` specifies a matrix of initial values. Unlike the usual Stata default, the column names for this matrix are assumed to correspond to those given in the variable list and need not be specified before the matrix is supplied. Any column names supplied will be ignored.

`lf(`*function*`)` specifies the likelihood function to be used. It is assumed to be of the `deriv0` type as specified in the Stata manual.

`iterate(`*#*`)` specifies a maximum number of iterations to compute before the program gives up. The default is 16,000.

`noconstant` specifies that there will not be a constant term in the estimation (the default is to include one). If a constant term is specified, an initial value for it must be specified as the last column of the matrix of initial values.

`step(`*#*`)` specifies how finely the program looks along the simplex path for higher likelihood values. A step value close to zero causes the program to look more frequently, while a value close to one causes it to only look once. The default is 0.1. Choosing finer values will probably only increase computational time per iteration, however if the function is non-concave in small intervals, a smaller value for step may pick up peaks that would otherwise be missed.

`span(`*#*`)` The default initial simplex consists of the vector specified in the `from` option, plus a vector corresponding to each parameter which is the same as the `from` vector, except that the initial value for that parameter in these vectors is the value specified in `from` plus that specified in `span`. For example, if there are two parameters, and the user specifies `from` as (1,1), the program will use the three points (1,1), (1+`span`,1), and (1,1+`span`) as the initial simplex. The default is 0.1.

`tolerance(`*#*`)` specifies how close the different simplex points must be in terms of the largest norm difference before the program considers them essentially the same and stops working and reports its estimates. The default is 0.00001.

`trace` outputs more information about the path of the simplex than anyone probably wanted to know.

## Remarks

For most likelihood-maximization problems, the standard routine supplied with the Stata package should be used. This standard routine is described in detail in section [6m] ml of the *Stata Reference Manual*, and this text assumes that the reader is familiar with the material in that section. It is based on the Newton–Raphson method, which uses the first and second derivatives of the likelihood function to perform the maximization. However, for some problems, the likelihood function may not be differentiable. In such cases, the standard routine will generally be able to approximate the slope of the likelihood function and will attempt a maximization, but these approximations may be inaccurate, and the reported maximum may be wrong. The simplex method, by contrast, will be consistent even if the likelihood function is not differentiable, as long as it is strictly globally concave. The tradeoff is that the simplex method is generally much slower than derivative-based methods. Therefore it should be used only when necessary.

The simplex method works as follows: For a likelihood function with $k$ parameters, the routine takes as input $k+1$ points (each of dimension $k$) as starting values for the estimates. These are similar to the starting values for the Newton–Raphson method, except that there are $k+1$ of them instead of just one. In the routine below, the user only specifies one such point, and the computer supplies the other $k$ by moving out 0.1 (or another specified number) along each axis. For example, suppose that in a three-dimensional parameter space, the user supplies the starting point $(1,1,1)$. The computer will then supply the additional three points $(1.1,1,1)$, $(1,1.1,1)$, and $(1,1,1.1)$. The program will then create a $(k+1)$-simplex out of these points in the parameter space (a simplex is a polygon; for example, a 3-simplex is a triangle, a 4-simplex is a quadrilateral, etc.).

An analogy may be helpful for developing an intuition how the simplex method works: Suppose that the parameter space is two-dimensional, so that the likelihood function looks like a hill, where each point on the ground represents a point in the parameter space and the height of the hill represents the value of the function at that point. By our assumption of strict concavity, the hill is never completely flat or completely vertical. An eccentric engineer decides to build a device to climb the hill out of three rods put together in a triangle (i.e., a 3-simplex) that lies on the ground and rolls. At each vertex of the triangle, there is a sensor that indicates how high that vertex is sitting on the hill. At any given point on the hill, one sensor will find that it is lower than (at least one of) the other two. That sensor springs up and, while the other two vertices stay still, the device rolls over on the edge opposite that vertex. This process is then repeated until the device fumbles up to the top of the hill. This is the basic idea behind the simplex method.

The careful reader will have noted that such a device will not work for every hill; indeed, the simplex is a bit more complicated than the device described above. For example, suppose we have a bird's eye view of the hill, looking straight down; let the $*$'s below indicate vertices, and let the one on the right be the lowest vertex.

```
                             *
             o-------------------*
                             *
```

As soon as the program finds that the vertex on the right is lower than the other two, it looks along the dotted line segment above. This segment is formed by convex combinations of the low vertex and the center of the convex hull of the remaining $k$ points. In order to get the segment to move out the other way, the convexity parameter varies between 1 and $-1$ instead of 1 and 0. It checks several (the default is twenty) evenly-spaced points on that line segment, finding the likelihood value at each point. It re-positions the vertex at whichever of those points has the highest likelihood value, and a new simplex is thus created and the process repeats itself. If none of the points on the line segment is higher than the original vertex (this happens more and more as the simplex approaches the maximum), then the whole simplex is flipped around the highest vertex and shrunk in half. For example, if "`a`" represents the highest vertex, then

```
                     a
                             b
                     c
```

becomes

```
                     c
                b
                     a
```

and the process then repeats itself. The program stops when the maximum distance between any of the two vertices (as measured by the Euclidean norm of their difference) is less than a specified value (the default is 0.00001). The value of the highest vertex is then reported.

Another disadvantage to the simplex method is that, because of the lack of derivatives, the Cramer–Rao theorem may not be used to derive a covariance matrix for the estimators. There is no ideal means of overcoming this difficulty. Bootstrapping may be used for most applications; for some, there may be an analytical solution to the covariance matrix that is specific to the problem at hand.

## Example

We will now illustrate the command by looking at an application with a non-differentiable maximand: a Least Absolute Deviation (LAD) estimator for a regression with a truncated dependent variable (as such, the maximand, or rather minimand, is technically not a likelihood function except in improbable situations; however, the maximizing technique is the same). Truncated variables are ones for which, if they are below a certain value, that value is reported instead of the true value of the variable (see [5s] cnreg). For example, if a variable $y$ is truncated below at 3, then any value measured below 3 (say 2.5) would be listed as 3 in the data set. Thus, supposing that for some estimation purpose, the "true" model we would like to estimate is

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon, \tag{1}$$

The data we have can only estimate the model

$$\mathbf{Y}_0 = \mathbf{X}\beta + \epsilon, \tag{2}$$

where $y_0 = y$ if $y \geq t$ and $y_0 = t$ if $y < t$. In general, OLS will be an inconsistent estimator of $\beta$, since the conditional expectation of $\epsilon$ given $x$ is greater than zero. If $y$ is normally distributed, then this model may be consistently estimated by the `tobit` command (see again [5s] cnreg). However, assuming normality when there is no a priori reason to—as many practitioners do—will lead to inconsistent estimates and false confidence in the standard errors reported.

Powell (1984) offers an alternative estimator that is consistent for all models that have non-skew-distributed error terms (i.e., in the equation above, $\epsilon$ has both mean and median zero). This estimator is based on minimizing the absolute value of the residuals, that is,

$$\widehat{\beta} = \min_{\beta} \left| y_0 - \max(x\beta, 0) \right|$$

This minimand can be minimized in the usual manner by maximizing its negative. While this will only be a true likelihood function if the error term has a Cauchy distribution (this tends to happen in real-life problems with probability zero), it can nevertheless be maximized with the same routines. The following program, `crlad`, is a routine to maximize such a function, and it is of the form `deriv0` as described in section [6m] ml of the *Stata Reference Manual*:

```
program define crlad
        version 4.0
        local b "`1´"
        local f "`2´"
        tempvar sad prdn
        matrix score `prdn´ = `b´
        qui replace `prdn´ = max(`prdn´,$ S_trunc)
        qui gen `sad´ = sum(sqrt(($ S_mldepn - `prdn´)2̂))
        scalar `f´ = `sad´[_N]
end
```

The macro `S_trunc` needs to be pre-programmed with the value at which the dependent variable is truncated; otherwise, the code largely mimics the routine `logitd0` explained in the above section of the manual.

To illustrate the problem, I used data from the census of manufactures (available from ftp://nber.harvard.edu/pub/productivity) to estimate a production function for U.S. manufacturing industries in 1988. If an economy exhibits constant returns to scale, the marginal productivity of capital and labor (which, in this case, coincide with the labor and capital shares of output) can be measured by a Cobb–Douglas function,

$$\text{output} = \text{capital}^{\beta_1} * \text{labor}^{\beta_2} * \exp(\epsilon)$$

or, in logs,

$$\ln(\text{output}) = \beta_1 * \ln(\text{capital}) + \beta_2 * \ln(\text{labor}) + \epsilon.$$

This regression is performed with the following fairly typical results:

```
. regress lnoutput capital labor
  Source |       SS       df       MS                  Number of obs =     450
---------+------------------------------               F(  2,   447) = 2481.14
   Model |  738.426806     2  369.213403               Prob > F      =  0.0000
Residual |  66.5171459   447  .148807933               R-squared     =  0.9174
---------+------------------------------               Adj R-squared =  0.9170
   Total |  804.943952   449  1.79274822               Root MSE      =  .38576
------------------------------------------------------------------------------
lnoutput |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 capital |   .3815745   .0256367     14.884   0.000        .331191     .431958
   labor |   .6502273   .0280634     23.170   0.000       .5950748    .7053799
   _cons |   .8045012   .0947638      8.490   0.000       .6182632    .9907392
------------------------------------------------------------------------------
```

The coefficient on labor is just under 2/3, that on capital is just over 1/3, and, while the hypothesis of constant returns to scale (capital + labor = 1) is rejected by an $F$ test (not shown here), the returns are not too far from constant.

We now ask the following question: What happens when we artificially truncate the dependent variable?

```
. summarize lnoutput
Variable |     Obs        Mean   Std. Dev.       Min        Max
---------+-----------------------------------------------------
lnoutput |     450    7.059864   1.338935   .1366132   10.64502
```

As can be seen above, the dependent variable has a mean of about seven. What happens if we truncate is at, say, five?

```
. generate ytrunc = max(lnoutput,5)
```

The OLS regression results become far off:

```
. regress ytrunc capital labor
  Source |       SS       df       MS                  Number of obs =     450
---------+------------------------------               F(  2,   447) = 1732.19
   Model |  602.127907     2  301.063954               Prob > F      =  0.0000
Residual |  77.6910396   447  .173805458               R-squared     =  0.8857
---------+------------------------------               Adj R-squared =  0.8852
   Total |  679.818947   449  1.51407338               Root MSE      =   .4169
```

```
------------------------------------------------------------------------------
  ytrunc |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 capital |   .4106052   .0277065     14.820   0.000       .3561541    .4650563
   labor |    .516488    .030329     17.029   0.000       .4568828    .5760932
   _cons |   1.389535   .1024145     13.568   0.000       1.188262    1.590809
------------------------------------------------------------------------------
```

The results from a `tobit` estimation are better:

```
. tobit ytrunc capital labor , ll

Tobit Estimates                                        Number of obs =    450
                                                       chi2(2)       =1082.19
                                                       Prob > chi2   = 0.0000
Log Likelihood =  -204.9773                            Pseudo R2     = 0.7253

------------------------------------------------------------------------------
  ytrunc |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 capital |    .402145   .0268302     14.989   0.000       .3494163    .4548736
   labor |   .6175567   .0302057     20.445   0.000       .5581942    .6769191
   _cons |   .8459794    .105341      8.031   0.000       .6389555    1.053003
---------+--------------------------------------------------------------------
     _se |   .3802426   .0131379             (Ancillary parameter)
------------------------------------------------------------------------------

Obs. summary:        30 left-censored observations at ytrunc<=5
                    420 uncensored observations
```

Here, at least, the true sample coefficients are within the 95% confidence interval. Now, let's try using the LAD estimator above:

```
. global S_trunc = 5

. matrix define b = (.5 .5 1)

. simplex ytrunc capital labor , from(b) lf(crlad)

Iteration 1: Minimum likelihood -391.5854797363281, maximum norm is .14142136
 output omitted
Iteration 70: Minimum likelihood -117.6286392211914, maximum norm is .00007363
Iteration 71: Minimum likelihood -117.6286239624023, maximum norm is .00007363
Iteration 72: Minimum likelihood -117.6286010742188, maximum norm is .00002775
Nonconcavity encountered.  Flipping and shrinking simplex.
Iteration 73: Minimum likelihood -117.6286010742188, maximum norm is .00001387
Iteration 74: Minimum likelihood -117.6285934448242, maximum norm is .00001383
Nonconcavity encountered.  Flipping and shrinking simplex.

Estimate using simplex, tolerance .00001:
Likelihood value  -117.6285934448242
No. Observations  450
------------------------------------------------------------------------------
  ytrunc |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 capital |   .3532639          .          .       .              .           .
   labor |   .6401788          .          .       .              .           .
   _cons |    1.02862          .          .       .              .           .
------------------------------------------------------------------------------
```

At each iteration, the routine reports the likelihood value of the "lowest" vertex, and the maximum distance between any two vertices. Here, the estimator converged after 74 iterations. This is actually a bit on the quick side; it is not uncommon for the estimator to require 1,000 iterations to converge. As can be seen, the results are better still, though, as explained above, there is no readily available means for estimating standard errors (although, for the particular LAD estimator exhibited here, Powell (1984) provides a generally consistent formula for the covariances of the parameter estimates, we have not built such an estimator into this routine).

Let's see what happens when we try to use the built-in Stata optimization techniques:

```
. matrix define b0 = (.5 .5 1)

. matrix colnames b0 = capital labor _cons

. matrix define b2 = (.5 .5 1)

. matrix colnames b2 = capital labor _cons

. ml begin

. ml function crlad

. ml method deriv0
. eq ytrunc capital labor

. ml model b2 = ytrunc , from(b0)

. ml sample mysamp

. ml maximize f V
Iteration 0:  Log Likelihood =  -142.2673
Iteration 1:  Log Likelihood = -121.02605
(nonconcave function encountered)
Iteration 2:  Log Likelihood =   -120.971
Iteration 3:  Log Likelihood = -119.84387
(unproductive step attempted)
Iteration 4:  Log Likelihood =  -119.3772
Iteration 5:  Log Likelihood =  -119.3772

. ml post treg

. ml mlout treg
                                                     Number of obs    =     450
                                                     Model chi2(2)    =       .
                                                     Prob > chi2      =       .
Log Likelihood =   -119.3771973
------------------------------------------------------------------------------
  ytrunc |      Coef.   Std. Err.       z     P>|z|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
 capital |   .4453788   5.43e-07        .     0.000       .4453778    .4453799
   labor |   .5504634          .        .       .              .           .
   _cons |   .8895498          .        .       .              .           .
------------------------------------------------------------------------------
```

As can be seen, although the estimator didn't get too far from the maximum (looking at the above output, we might surmise that this is more due to the estimator's built-in error-correcting routines than anything else), it didn't get terribly close, either . The second derivative of the sample error function should have been zero where it existed, and hence the routine was mostly unable to compute estimates of the parameter covariance matrix. Unlike the simplex routine, however, this one converged within a few iterations, which is why it is a preferable routine when it does work. The above example has attempted to illustrate the following: (1) There are estimations for which the maximand is not differentiable; (2) Stata's built-in maximization routines, because they depend on differentiability, are generally inconsistent in these cases, whereas the simplex routine is consistent; (3) The simplex routine is very slow and should only be used when necessary.

## References

Himmelblau, D. M. 1972. *Applied Nonlinear Programming.* New York: McGraw–Hill.

Nelder, J.A., and R. Mead. 1965. A simplex method for function minimization. *The Computer Journal* 7: 308–313.

Powell, J. 1984. Least absolute deviations estimation for the censored regression model. *Journal of Econometrics* 25: 303–325.

## snp11 Test for trend across ordered groups revisited

Peter D. Sasieni, Katarzyna A. Stepniewska, and Douglas G. Altman
Imperial Cancer Research Fund, London, FAX (011)-44-171-269-3429

**nptr** provides a nonparametric test of trend in a variable **varname** across ordered groups defined by **groupvar** (Stepniewska and Altman, 1992).

The new version incorporates an improved algorithm (over the algorithm used in Stata's **nptrend** command) which should make it run considerably faster. The old **score(***scorevar***)** option has been replaced by two new options as explained below. There is also a new immediate command **nptri**, which emphasizes that this nonparametric trend test may be used with two-way tables of frequencies when both variables have ordered categories.

## Syntax

**nptr** *varname* [ **if** *exp* ] [ **in** *range* ] **, by(***groupvar***)** [ **ordinal midrank nodetail** ]

**nptri** $\#_{11}$ $\#_{12}$ [...] \ $\#_{21}$ $\#_{22}$ [...] [\ ...] [**, replace midrank** ]

## Options

**ordinal** uses the values 1, 2, 3 ... instead of the values in *groupvar* as the scores in the test.

**midrank** uses the midranks of *groupvar* instead of the values in *groupvar* as the scores in the test. This is equivalent to ridit scores. Note that **midrank** is an alternative to **ordinal**.

`nodetail` suppresses the sum of ranks output and displays the test statistic only.

`replace` indicates that the immediate data specified as arguments to `tabi` command are to be left as the current data.

The default scoring uses the values in *groupvar*. There was nothing in the old version of `nptrend` to make sure that there was a one-to-one correspondence between the values in *groupvar* and those in *scorevar*. Whenever there was a one-to-one correspondence, the same answer could be obtained by using *scorevar* in the `by` option. The new options allow two nonparametric replacements for the explicit values in *groupvar*. The first uses codes 1, 2, 3, ... and is the default in the immediate version. The other uses the midranks after sorting the data according to *groupvar*.

When *groupvar* has only two values, all the scoring options are equivalent. The test is then equivalent to the Wilcoxon (Mann–Whitney) rank-sum test, which is itself equivalent to the Kruskal–Wallis test when there are just two groups.

`nptr` can also be used with two-way contingency tables when both variables are ordinal. These tests can then be viewed as alternatives to those based on gamma and Kendall's $\tau_b$ provided as options by the `tabulate` command. Unless the `midrank` option is used, the `nptr` test does not treat the two variables symmetrically. It assumes that *varname* is dependent on *groupvar*. In the immediate version the rows are treated as the dependent variable.

## Examples

The data relates the number of dysplastic naevi (atypical moles) on a patient to the thickness of their melanoma (in mm). Both the number of dysplastic naevi and the thickness of the melanoma have been grouped: 0, 1, 2–3, and 4 or more for moles; and 0–0.75 mm, 0.76–1.5 mm, 1.51–3.0 mm and greater than 3 mm. First, we tabulate the data and calculate tests based on gamma and Kendall's $\tau_b$.

```
. tab Thick N_dn, gamma taub
           | N_dn
      Thick|        0          1         2-3        4+ |     Total
-----------+-------------------------------------------+----------
       .75 |      104         12         11         21 |       148
       1.5 |       62          8          7          1 |        78
         3 |       26          3          5          6 |        40
         9 |       27          5          2          0 |        34
-----------+-------------------------------------------+----------
      Total|      219         28         25         28 |       300
                    gamma =  -0.1086  ASE = 0.097
          Kendall´s tau-b =  -0.0583  ASE = 0.051
. di "  z-score for gamma = " %5.2f _result(10)/_result(11)
  z-score for gamma = -1.12
. di "  z-score for tau-b = " %5.2f _result(12)/_result(13)
  z-score for gamma = -1.13
```

Note that the tests based on gamma and $\tau_b$ have the same efficacy for testing independence. Hence, in practice, they will usually be in close agreement. Next, we illustrate the use of `nptr` treating the thickness of the melanoma as the dependent variable.

```
. nptr Thick, by(N_dn)
      N_dn     score       obs      sum of ranks
       0.0       0.0       219       33436.5
       1.0       1.0        28        4551.0
       3.0       3.0        25        3931.5
      10.0      10.0        28        3231.0
     z  = -2.21,  chi-squared(1) =   4.87
  P>|z| = 0.0273
```

The $z$ statistic based on the weighted Wilcoxon test is nearly twice as large as that based on Kendall's $\tau_b$. Note however that the scoring used is rather extreme—the group with four or more dysplastic naevi are given score 10 compared to scores 0, 1 and 3 for the other three groups. Next we investigate this by using equally spaced scores.

```
. nptr Thick, by(N_dn) ordinal
      N_dn     score       obs      sum of ranks
       0.0       1.0       219       33436.5
       1.0       2.0        28        4551.0
       3.0       3.0        25        3931.5
      10.0       4.0        28        3231.0
     z  = -1.65,  chi-squared(1) =   2.74
  P>|z| = 0.0981
```

This has indeed reduced the significance of the association between thickness and number of dysplastic naevi. Finally we assign scores based on the midranks.

```
. nptr Thick, by(N_dn) midrank
```

```
        N_dn     score       obs      sum of ranks
         0.0     110.0       219       33436.5
         1.0     233.5        28        4551.0
         3.0     260.0        25        3931.5
        10.0     286.5        28        3231.0

       z  = -1.13,  chi-squared(1) =   1.28
    P>|z| = 0.2574
```

It is merely a coincidence that in this example the $z$ statistic using midranks is equal to that based on Kendall's $\tau_b$. Note that the Wilcoxon test using midranks to score the groups is similar to using Kendall's $\rho_b$ the analog of Spearman's rank correlation.

Next we treat the number of dysplastic naevi as the dependent variable and perform the three different Wilcoxon tests.

```
  . nptr N_dn, by(Thick)
       Thick     score       obs      sum of ranks
         .75       .75       148       23118.5
         1.5       1.5        78       10794.5
         3.0       3.0        40        6579.5
         9.0       9.0        34        4657.5

       z  = -1.08,  chi-squared(1) =   1.18
    P>|z| = 0.2779
  . nptr N_dn, by(Thick) ordinal
       Thick     score       obs      sum of ranks
         .75       1.0       148       23118.5
         1.5       2.0        78       10794.5
         3.0       3.0        40        6579.5
         9.0       4.0        34        4657.5

       z  = -1.00,  chi-squared(1) =   0.99
    P>|z| = 0.3190
```

Note that when we use the midranks, we get the same result whichever way round we analyze the data. gamma and $\tau_b$ are also symmetric in this sense.

```
  . nptr N_dn, by(Thick) midrank
       Thick     score       obs      sum of ranks
         .75      74.5       148       23118.5
         1.5     187.5        78       10794.5
         3.0     246.5        40        6579.5
         9.0     283.5        34        4657.5

       z  = -1.13,  chi-squared(1) =   1.28
    P>|z| = 0.2574
```

`nptri` is illustrated with data on the number of strains of *Staphylococcus aureus*, isolated from patients in a certain hospital between 1947 and 1950, which were resistant or sensitive to 1 unit per cc of penicillin (data from Armitage 1971, 383).

```
  . nptri 45 41 113 53 \ 194 145 185 107
       z  = -4.57, chi-squared(1) =  20.93
    P>|z| = 0.0000
  . nptri 45 41 113 53 \ 194 145 185 107,mid
       z  = -4.60, chi-squared(1) =  21.16
    P>|z| = 0.0000
  . nptri 45 194 \ 41 145 \ 113 185 \ 53 107
       z  = -4.60, chi-squared(1) =  21.16
    P>|z| = 0.0000
```

For completeness, these $z$ statistics may be compared to those obtained from other analyses: 4.58 for Armitage's trend test and 4.60 for logistic regression.

For an in-depth discussion of the related statistical issues, including properties of the measures of association gamma and $\tau_b$, and guidelines regarding category choice and comparison between different approaches for analyzing ordinal-ordinal tables; see Agresti 1984.

## Also see

[5s] kwallis, [5s] nptrend, [5s] signrank, [5s] spearman

## References

Agresti, A. 1984. *Analysis of Ordinal Categorical Data.* New York: John Wiley & Sons.

Armitage, P. 1971. *Statistical Methods in Medical Research.* Oxford: Blackwell Scientific Publications.

Stepniewska, K. A. and D. G. Altman. 1992. snp4: Non-parametric test for trend across ordered groups. *Stata Technical Bulletin* 9: 21–22.

## sts11 Hildreth–Lu regression

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

The syntax of `hlu` is

`hlu` [ *depvar* [ *varlist* ] ] [ `if` *exp* ] [ `in` *range* ] [ , `level(`*#*`)` `nolog` ]

The two principal remedial measures for dealing with serially correlated data in regression are to add one or more independent variables to the model or to use transformed variables. Stata already includes the means for transforming the variables by using the Cochrane–Orcutt procedure. The Hildreth–Lu procedure (Neter, et al., 500) is a closely related procedure that has been shown to work better for small sample sizes. Both procedures must estimate the autocorrelation parameter $\rho$. The Cochrane–Orcutt procedure estimates this by assuming a functional form for the error term in the proposed model. The Hildreth–Lu procedure uses an iterative technique that guarantees that the error sum of squares for the transformed model, given below, is minimized.

$$\text{SSE} = \sum_t \left( Y_t' - \widehat{Y}_t' \right)^2$$

The procedure for calculating the correlation parameter for use in the regression transformations is similar to the search for the parameter $\lambda$ in the power transformation of Box–Cox. The value chosen for $\rho$ is the one which minimizes the error sum of squares for the transformed regression model. The transformation of the regression model is given by

$$Y_t' = Y_t - \rho Y_{t-1}$$
$$X_t' = X_t - \rho X_{t-1}$$

The motivation for using the Hildreth–Lu procedure over the Cochrane–Orcutt is that the Cochrane–Orcutt tends to underestimate the parameter $\rho$. When this bias is serious, it may significantly reduce the effectiveness of the Cochrane–Orcutt procedure. The Hildreth–Lu procedure does not require any iterations once the autocorrelation parameter is estimated.

The algorithm for finding the parameter $\rho$ is a simple bisection search that continues until the estimate does not differ until the fifth decimal place (meaning there are 15 maximum iterations).

As in the Cochrane–Orcutt procedure, one must `gen _iter=1` before you will be able to use the `predict` command.

### Example

Data are available for industry sales (in millions of dollars) and we would like to use that information to model the company sales for company X. These sales figures are quarterly sales for five years.

```
. reg csales isales
  Source |       SS       df       MS                  Number of obs =      20
---------+------------------------------               F(  1,    18) =14888.15
   Model |  110.256901     1  110.256901               Prob > F      =  0.0000
Residual |  .133302302    18  .007405683               R-squared     =  0.9988
---------+------------------------------               Adj R-squared =  0.9987
   Total |  110.390204    19  5.81001072               Root MSE      =  .08606

------------------------------------------------------------------------------
  csales |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  isales |   .1762828   .0014447    122.017   0.000       .1732475    .1793181
   _cons |  -1.454753   .2141461     -6.793   0.000      -1.904657   -1.004849
------------------------------------------------------------------------------
```

Noting that we have serial correlation in the errors, we first attempt to correct for that using the Cochrane–Orcutt procedure.

```
. corc csales isales, nolog
(Cochrane-Orcutt regression)
  Source |       SS       df       MS                  Number of obs =      19
---------+------------------------------               F(  1,    17) =  596.10
   Model |  2.51740536     1  2.51740536               Prob > F      =  0.0000
Residual |  .071793235    17  .004223131               R-squared     =  0.9723
---------+------------------------------               Adj R-squared =  0.9706
   Total |   2.5891986    18  .143844367               Root MSE      =  .06499
```

```
------------------------------------------------------------------------------
  csales |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  isales |   .1615211   .0066156     24.415   0.000       .1475634    .1754788
   _iter |   1.307525   1.267446      1.032   0.317      -1.366553    3.981602
------------------------------------------------------------------------------
     rho |     0.9441     0.0229     41.316   0.000         0.8961      0.9921
------------------------------------------------------------------------------
```

We know that the Cochrane–Orcutt procedure tends to underestimate the correlation and the correlation is very high in this problem. Therefore we may also look at the Hildreth–Lu approach:

```
. hlu csales isales, nolog
(Hildreth-Lu regression)
  Source |       SS       df       MS                  Number of obs =      19
---------+------------------------------               F(  1,    17) =  553.13
   Model |  2.33192386     1  2.33192386               Prob > F      =  0.0000
Residual |  .071670371    17  .004215904               R-squared     =  0.9702
---------+------------------------------               Adj R-squared =  0.9684
   Total |  2.40359423    18  .133533013               Root MSE      =  .06493

------------------------------------------------------------------------------
  csales |      Coef.   Std. Err.       t     P>|t|       [95% Conf. Interval]
---------+--------------------------------------------------------------------
  isales |   .1605229   .0068254     23.519   0.000       .1461227    .1749232
   _iter |   1.739166   1.432765      1.214   0.241      -1.283703    4.762035
------------------------------------------------------------------------------
     rho |     0.9588
------------------------------------------------------------------------------
```

A comparison of the Hildreth–Lu, Cochrane–Orcutt, and Prais–Winsten methods for dealing with regression with autocorrelated error terms essentially provide the same estimate of $\sigma^2$ (the variance of the disturbance terms). The estimated standard error of the $\beta$ parameters from ordinary least squares regression may be seriously underestimated when positive autocorrelation is present.

## Saved Results

The `hlu` command stores the command name in `S_E_cmd` and the name of the dependent variable in `S_E_depv`. In addition, `hlu` saves in the macro `S_E_rho` the estimate of rho.

## References

Neter, J., W. Wasserman and M. H. Kutner. 1989. *Applied Regression Models*. Homewood, IL: Irwin.

## STB categories and insert codes

Inserts in the STB are presently categorized as follows:

| | | | |
|---|---|---|---|
| *General Categories* | | | |
| *an* | announcements | *ip* | instruction on programming |
| *cc* | communications & letters | *os* | operating system, hardware, & interprogram communication |
| *dm* | data management | *qs* | questions and suggestions |
| *dt* | datasets | *tt* | teaching |
| *gr* | graphics | *zz* | not elsewhere classified |
| *in* | instruction | | |
| *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 sample |
| *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<br>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<br>D-42697 Solingen<br>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<br>113 55 Stockholm<br>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<br>75009 Paris<br>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<br>Postbox 220<br>5150 AE Drunen<br>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<br>West Wickham<br>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,<br>N°13–9° Dto. Quinta Grande<br>2720 Alfragide Portugal |
| Phone: | +351 (01) 4719337 |
| Telemóvel: | 0931 62 7255 |
| Email: | 100412.2603@compuserve.com |
| Countries served: | Portugal |